Linköping Studies in Science and Technology

Dissertation No. 1274

Algorithms and Hardness Results for Some Valued CSPs

by

Fredrik Kuivinen

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2009

To Linnea

Abstract

In the Constraint Satisfaction Problem (CSP) one is supposed to find an assignment to a set of variables so that a set of given constraints are satisfied. Many problems, both practical and theoretical, can be modelled as CSPs. As these problems are computationally hard it is interesting to investigate what kind of restrictions of the problems implies computational tractability. In this thesis the computational complexity of restrictions of two optimisation problems which are related to the CSP is studied. In optimisation problems one can also relax the requirements and ask for an approximatively good solution, instead of requiring the optimal one.

The first problem we investigate is Maximum Solution (Max Sol) where one is looking for a solution which satisfies all constraints and also maximises a linear objective function. The Maximum Solution problem is a generalisation of the well-known integer linear programming problem. In the case when the constraints are equations over an abelian group we obtain tight inapproximability results. We also study Max Sol for so-called maximal constraint languages and a partial classification theorem is obtained in this case. Finally, Max Sol over the boolean domain is studied in a setting where each variable only occurs a bounded number of times.

The second problem is the Maximum Constraint Satisfaction Problem (Max CSP). In this problem one is looking for an assignment which maximises the number of satisfied constraints. We first show that if the constraints are known to give rise to an **NP**-hard CSP, then one cannot get arbitrarily good approximate solutions in polynomial time, unless $\mathbf{P} = \mathbf{NP}$. We use this result to show a similar hardness result for the case when only one constraint relation is used. We also study the submodular function minimisation problem (SFM) on certain finite lattices. There is a strong link between Max CSP and SFM; new tractability results for SFM implies new tractability results for Max CSP. It is conjectured that SFM is the only reason for Max CSP to be tractable, but no one has yet managed to prove this. We obtain new tractability results for SFM on diamonds and evidence which supports the hypothesis that all modular lattices are tractable.

Sammanfattning

I ett villkorsprogrammeringsproblem är uppgiften att tilldela värden till variabler så att en given mängd villkor blir uppfyllda. Många praktiska problem, så som schemaläggning och planering, kan formuleras som villkorsprogrammeringsproblem och det är därför önskvärt att ha effektiva algoritmer för att hitta lösningar till denna typ av problem.

De generella varianterna av dessa problem är **NP**-svåra att lösa. Detta innebär att det antagligen inte finns effektiva algoritmer för problemen (om inte $\mathbf{P} = \mathbf{NP}$ vilket anses vara mycket osannolikt). Av denna anledning förenklar vi problemet genom att studera restriktioner av det och ibland nöjer vi oss med approximativa lösningar.

I den här avhandlingen studeras två varianter av villkorsprogrammeringsproblemet där man inte bara ska hitta en lösning utan hitta en så bra lösning som möjligt. I den första varianten är målet att hitta en tilldelning där samtliga villkor uppfylls och att en viktad summa av variablerna maximeras. Detta problem kan ses som en generalisering av det välkända linjära heltalsprogrammeringsproblemet. I den andra varianten är målet att hitta en tilldelning som uppfyller så många villkor som möjligt.

Då det gäller den första varianten, då man ska hitta en lösning som uppfyller samtliga villkor som också maximerar summan av variablerna, presenteras nya resultat för ett antal specialfall. De så kallade maximala villkorsmängderna studeras och komplexiteten för ett antal av dessa bestäms. Vi studerar också en variant av problemet över den Boolska domänen då antal variabelförekomster är begränsat. I detta fall ger vi en partiell klassifikation över vilka villkorsmängder som är hanterbara och vilka som inte kan lösas effektivt.

För den andra varianten, då man ska uppfylla så många villkor som möjligt, presenteras några nya effektiva algoritmer för vissa restriktioner. I dessa algoritmer löses det mer generella problemet av minimering av submodulära funktioner över vissa ändliga latticar. Vi bevisar också ett resultat som beskriver exakt när det finns effektiva algoritmer då man endast har tillgång till en typ av villkor.

Acknowledgements

First and foremost I would like to thank my supervisor Peter Jonsson for all the support he has given me during my time as a graduate student. In particular, the fact that Peter always has had time to discuss research questions has been a tremendous resource.

I also want to thank:

- my co-authors Gustav Nordh, Peter Jonsson, and Andrei Krokhin;
- Tommy Färnqvist for reading an earlier version of this manuscript and pointing out lots of errors;
- my past and present colleagues at the theoretical computer science laboratory deserve a huge thanks for providing a nice environment to work in;
- Anton Blad and Jakob Rosén for all the interesting discussions, TRUT, and the geese feeding;
- all my friends for making life fun;
- my family for always supporting me;
- and finally, Linnea for all her love and encouragement and for reminding me that there is more to life than research.

This research work was funded in part by CUGS (the National Graduate School in Computer Science, Sweden).

List of Papers

Parts of this thesis are based on the following publications.

- F. Kuivinen. Tight approximability results for the maximum solution equation problem over $\mathbb{Z}_p$. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS '05)*, pages 628–639. Springer, 2005.

- F. Kuivinen. Approximability of bounded occurrence max ones. In *Proceedings of the 31th International Symposium on Mathematical Foundations of Computer Science (MFCS '06)*, pages 622–633. Springer, 2006.

- P. Jonsson, A. Krokhin, and F. Kuivinen. Hard constraint satisfaction problems have hard gaps at location 1. *Theor. Comput. Sci.*, 410(38–40):3856–3874, 2009.

  A conference version of this article first appeared as:

  - P. Jonsson, A. Krokhin, and F. Kuivinen. Ruling out polynomial-time approximation schemes for hard constraint satisfaction problems. In *Proceedings of the Second International Symposium on Computer Science in Russia (CSR '07)*, pages 182–193. Springer, 2007.

- P. Jonsson, F. Kuivinen, and G. Nordh. Max ones generalized to larger domains. *SIAM J. Comput.*, 38(1):329–365, 2008.

  A conference version of this article first appeared as:

  - P. Jonsson, F. Kuivinen, and G. Nordh. Approximability of integer programming with generalised constraints. In *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP '06)*, pages 256–270. Springer, 2006.

- F. Kuivinen. Submodular functions on diamonds. In *International Symposium on Combinatorial Optimization (CO '08)*, pages 53–54. 2008.

Contents

Part I

Introduction and Background

Chapter 1

Introduction

1.1 How Hard Is It to Colour a Map?

In this thesis we will study how hard it is to solve certain problems on ones computer. So what is a problem in this context? As a concrete example consider the following scenario: you are given a map of the countries of some world and are asked to colour the countries with two colours so that no two countries which shares a border have the same colour (see Figure 1.1 for one example of a map). The question is if there is such a "valid" colouring or not. There is a simple algorithm which solves this problem so that the number of elementary steps needed by the algorithm is bounded by $c \cdot n$, where n is the number of countries in the map and c is a constant independent of n. Indeed, to check if there is a valid colouring start with colouring an arbitrary country, A, with one of the colours—say black—and continue to colour the neighbours of A with the other colour—say white. These recently coloured neighbours of A will in turn force us to colour their neighbours black. We proceed in this manner until all countries are coloured or when we are forced to colour two neighbouring countries with the same colour. It is easy to convince oneself that this algorithm succeeds if and only if there is a valid colouring of the entire map.

If you are instead asked to colour the countries with three colours such that no two countries which shares a border have the same colour, then it seems that the problem gets significantly harder. In particular, the method used above to test if there is a valid colouring with two colours breaks down for three colours. The reason is that if we have coloured some country A, then there is not a unique valid colour we can assign to the neighbours of A. Despite much effort there is currently no known algorithm which solves this problem in time $c \cdot n^c$ for any fixed c. In fact, the current fastest algorithm has a running time of $c \cdot 1.33^n$, for some constant c. [11] In other words, there is no known algorithm with a running time bounded by a polynomial in the number of countries. The difference in the running times between these two

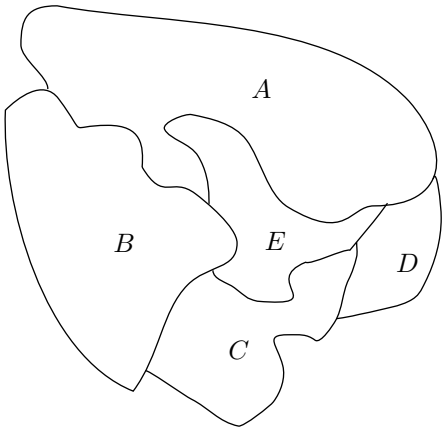


Figure 1.1: A map which can be coloured by three colours but not by two. One valid 3-colouring is obtained by colouring A and C red, B and D green, and E blue. The map is not 2-colourable as the three countries A, B, and E all share borders.

algorithms is enormous. For concreteness, assume that both algorithms can colour a map with 100 countries in one second on some computer. With the first algorithm it will then take no more than two seconds to colour a map with 200 countries. If we start running the second algorithm on a map with 200 countries we will have to be prepared to wait roughly 80 000 years for the result! Figure 1.1 contains a map which is colourable by three colours, but not by two.

The family of all problems which can be solved in polynomial time is called **P**. The 2-colourability problem above is contained in **P** as there is an algorithm to solve the problem whose running time is bounded by a polynomial in n. Another important family of problems is called **NP**. **NP** contains all problems such that if the answer is YES, then there is a *certificate* of this fact which can be verified efficiently. One example of a problem in **NP** is the 3-colourability problem. In this case the certificate is the colouring. It is clear that given a colouring we can easily check if it is valid or not efficiently, even though it may be very hard to *find* the colouring. It is easy to see that $\mathbf{P} \subseteq \mathbf{NP}$; if we can compute the answer efficiently we can certainly verify a claim of the form "the map is 2-colourable by this colouring" simply by ignoring the certificate (the colouring in this case) and compute the answer ourselves. One of the major open questions in complexity theory is if **P** equals **NP** or not. It is widely believed that **P** does not equal **NP**, but so far no one has managed to prove this.

Some of the problems in **NP** are at least as hard as any other problem in **NP**. These problems are said to be **NP**-*hard* and have the property that if any one of them is contained in **P**, then $\mathbf{P} = \mathbf{NP}$. If an **NP**-hard

problem is also contained in **NP**, then it is **NP**-*complete*. Hence, the **NP**-complete problems are the hardest problems in **NP**. The 3-colourability problem described above is known to be **NP**-complete. [66] So if one manage to construct an efficient algorithm which decides if a map is colourable with three colours or not, then one has proved that **P** is equal to **NP**. We do not currently know *why* we do not have a better algorithm for this problem. Is it because some inherent difficulty in the problem makes it impossible to construct an efficient algorithm? Or, is it our inability and lack of imagination which is the culprit? As mentioned above it is widely believed that **P** $\neq$ **NP** and hence that the former hypothesis is true.

Somewhat unexpectedly, most natural problems in **NP** are either in **P** or are **NP**-complete. It seems that problems of intermediate complexity (neither "easy", i.e., contained in **P**, nor hardest in **NP**, i.e., **NP**-complete) are far apart.[1] A large part of this thesis is devoted to show that certain problems are either **NP**-complete or contained in **P**.

One can also consider an optimisation variant of the 2-colouring problem. Instead of requiring that for every border the two countries sharing that border have different colours, we are asked to construct a colouring which maximises the number of borders where the two countries sharing the border have different colours. This problem is called Maximum Cut (the problem is equivalent to finding a cut of maximum size in a graph). Even though it is fairly easy to decide if the map is 2-colourable or not (that is, if we can colour the map so that no two neighbouring countries have the same colour) it seems to be much harder to solve Maximum Cut. Indeed, the problem is known to be **NP**-hard. [100]

A related problem, where one is instead asked to find a 2-colouring which *minimises* the number of borders where the two countries have different colours, is called Minimum Cut. (In this problem we also allow constraints of the form "this country has to be coloured black" and analogously for white, otherwise the problem becomes trivial—just colour every country with the same colour.) Minimum Cut can be solved in time polynomial in n by, for example, the max flow–min cut theorem and algorithms for maximum flow (see, e.g., [40]).

In this thesis we will study problems of the same general form as the examples of optimisation problems above: given some variables and some constraints we are asked to find an assignment to the variables (colours to the countries in the examples above) so that some function is maximised,

[1] There are a few candidates for natural problems of intermediate complexity. Three of them are graph isomorphism (given two graphs, are they isomorphic?), factoring (find the prime factors of an integer) and polynomial identity testing (given a polynomial, is it identically zero?). There is some evidence that none of these problems are **NP**-complete. Currently there is no known polynomial time algorithm for any of them.

Due to a theorem of Ladner [110] it is known that if **P** $\neq$ **NP**, then there are problems in **NP** of intermediate complexity. However, these problems have been constructed solely for the purpose of having this property and cannot be said to correspond to a "natural" computational problem.

for example, the number of satisfied constraints (borders with two countries coloured differently in the examples above). There are countless variants of these problems: one can allow k colours instead of two (k is an arbitrary positive integer), one can allow other types of constraints, for example, "if countries A, B and C shares borders then at least one of them should be coloured white", etc. At some point it is not sensible to talk about countries and colours any more—the problem is so different from the original one that the somewhat familiar concepts no longer make sense. In the next section we will define the *constraint satisfaction problem* which is a framework in which many of these types of problems can be formulated.

As we saw in the examples above, such problems are sometimes efficiently solvable (as in the case of finding 2-colourings) and sometimes there probably is no efficient algorithm (e.g., for finding 3-colourings). As a general theme in this thesis we will be interested in what kind of restrictions on the allowed constraints make the problems efficiently solvable.

In the few paragraphs above we have barely scratched the surface of the area known as computational complexity theory. We refer the reader to [6, 121, 136] for a much more thorough introduction to this research field.

1.2 Constraint Satisfaction Problems

In the *constraint satisfaction problem* (CSP) one is given a set of variables and a set of constraints on the variables and is asked to assign values from some domain to the variables so that all constraints are simultaneously satisfied. In this section we will define CSPs and give a couple of examples. For more comprehensive treatments of CSPs we refer the reader to [4, 129].

To define the CSP more formally we need to introduce some notation. Let D be a finite set and let n be a positive integer. The set of all n-tuples of elements from D is denoted by D^n. Any subset of D^n is an *n-ary relation* over D. The set of all finitary relations over D is denoted by R_D. The CSP can now be formulated as follows.

Definition 1.1 (Constraint Satisfaction Problem (CSP))**.** *The* constraint satisfaction problem *is defined to be the decision problem with instance* (V, D, C)*, where*

- V *is a set of variables;*
- D *is a set of values (the* domain*); and*
- C *is a multiset of constraints* $\{C_1, \ldots, C_q\}$*, in which each constraint* C_i *is a pair* $(\boldsymbol{s_i}, R_i)$*, where* $\boldsymbol{s_i}$ *is a list of variables of length* m_i*, called the* constraint scope*, and* R_i *is an* m_i*-ary relation over the set* D *called the* constraint relation*.*

The question is whether there exists a function from V *to* D *such that, for each constraint in* C*, the image of the constraint scope is a member of the constraint relation.*

	9			3	1		8	
			5			9		
			4			2	3	
3		8					1	6
	6	5		1		7	2	
1	2					8		4
	7	2			8			
		6			5			
	1		2	6			5	

Figure 1.2: A Sudoku puzzle. The blocks are the 3×3 square grids with thick borders.

The k-colourability problems from Section 1.1 are examples of CSPs.

Example 1.2 (k-colourability as a CSP)**.** *Let k be a positive integer. In the k-colourability problem we are given a graph $G = (V, E)$ and are asked if there is a valid k-colouring of V. A valid colouring is a function $c : V \to \{1, 2, \ldots, k\}$ such that if $v, u \in V$ are adjacent in G, then $c(u) \neq c(v)$. (This is the same definition as we used in Section 1.1.) We can see the k-colourability problem as a CSP with domain $\{1, 2, \ldots, k\}$, variables V, and one constraint of the form $u \neq v$ for each edge (u, v) in E. It is not hard to see that this CSP is equivalent to the k-colourability problem, that is, G is k-colourable if and only if the CSP instance has a solution.*

The puzzle *Sudoku* can also be seen as a CSP.

Example 1.3 (Sudoku as a CSP)**.** Sudoku *is a popular puzzle played on a square 9×9 grid. At the beginning some of the squares in the grid are filled with numbers from 1 to 9. The goal is to fill in numbers in the other squares with the restrictions that no row, column or block (some of the 3×3 squares) contain any duplicates. See Figure 1.2 for an example of a Sudoku.*

To model Sudoku as a CSP we need to introduce a coordinate system on the grid. We will use coordinates of the form (c, r) where c is the column and r the row. We start counting from 1 and $(1, 1)$ is the bottom left square in the grid. (So $(7, 4)$ is the square on the seventh column and fourth row. In Figure 1.2 this square contains 8.)

Sudoku puzzles can be modelled as CSPs with domain $\{1, 2, \ldots, 9\}$ and one variable $x_{(i,j)}$ for each coordinate (i, j). The only constraint relations we will need are the inequality constraint and the constant constraints. To impose the restriction that row r does not contain any duplicates we introduce the constraints $x_{(i,r)} \neq x_{(j,r)}$ for every $i, j \in \{1, 2, \ldots, 9\}, i \neq j$. Similarly, for column c we get the constraints $x_{(c,i)} \neq x_{(c,j)}$ for every $i, j \in$

$\{1, 2, \ldots, 9\}$, $i \neq j$. *To make sure that no block contains any duplicates we need to add the constraints*

$$x_{(3a+i,3b+j)} \neq x_{(3a+k,3b+l)}$$

for every $a, b \in \{0, 1, 2\}$ *and* $i, j, k, l \in \{1, 2, 3\}, (i, j) \neq (k, l)$. *(Here* (a, b) *can be seen as the coordinates of a* block *and* $(i, j), (k, l)$ *are coordinates* within *the block.) Finally we need to deal with the fact that some of the squares are given to us. This is done by introducing constant constraints of the form* $x_{(i,j)} = c$ *if square* (i, j) *has the number* c *in it. As an example, for the Sudoku in Figure 1.2 we need to add* $x_{(2,1)} = 1$, $x_{(4,1)} = 2$, $x_{(5,1)} = 6$, *and* $x_{(8,1)} = 5$ *for the squares on the bottom row.*

Any solution to this CSP is a solution to the Sudoku; and conversely any solution to the Sudoku is a solution to the CSP.

In Definition 1.1 above we allow multisets of constraints (a *multiset* is a set where the same element can appear several times). For CSP it does not make any difference to allow a multiset of constraints instead of only a set of constraints—if there are more than one constraint which has the same constraint scope and constraint relation, then they are either all satisfied or none of them are satisfied. However, in two of the problems introduced below (Max CSP and VCSP) the distinction between multisets and sets are significant. We do therefore allow multisets in our definition of CSP as well.

In this thesis we will only be interested in finite domains, hence D will always be a finite set. (CSPs with infinite domains have also been studied quite a lot, see, for example, the survey [14].) The general definition of constraint satisfaction problems found in Definition 1.1 is not so interesting from a complexity theoretical point of view because the problem is clearly hard to solve. For a fixed domain and explicitly represented relations it is **NP**-complete. An explicit example is the k-colourability problem presented above, which is **NP**-hard for $k \geq 3$. We will therefore be looking at restrictions of the general problem. When restricting the problem one loses some of the expressive power and, hopefully, gains computational tractability. In some applications this is not tolerable, maybe we really have a general 3-colourability problem we want to solve. The hope is that one often can impose some kind of restriction on the instances to make the problem easier to tackle. In this thesis we will therefore study a restriction on CSPs which is called *constraint language restrictions*. Before we delve into the definition of this kind of restriction some motivational remarks and notes on practical aspects are in order.

1.3 Why Study CSPs?

The art of solving CSPs is called *constraint programming*. Constraint programming is an intensively studied subject both from a theoretical point

of view, which this thesis is a small part of, and from a practical point of view. Indeed, there are entire conferences and journals devoted to the subject.[2] The practical part of constraint programming consists, among other things, of constructing efficient solvers and modelling real-world problems in the constraint programming framework. The natural question "Why study CSPs?" can be answered in at least two ways:

- constraint programming is flexible and can model many problems. It is widely used to solve real world problems in a variety of domains, for example, scheduling, planning, network design, and bioinformatics [129].

- Both noncommercial and commercial mature software systems for solving CSPs are available. The noncommercial ones include Gecode, Mozart/Oz, and ECL^{i}PS^e. Two commercial ones are Sicstus Prolog and ILOG CP.

The combination of the two bullet points above makes constraint programming a very useful programming paradigm. In this thesis only some theoretical aspects of CSPs will be investigated. We refer the reader to [4, 129] for detailed treatments on the more practical aspects of constraint programming.

1.4 Constraint Language Restrictions

One kind of restriction of the CSP which has been studied a lot is *constraint language restrictions*. In this type of restriction the types of the constraints that one is allowed to use is restricted. A *constraint language* over a finite set, D, is a finite set $\Gamma \subseteq R_D$ (recall that R_D is the set of all finitary relations over the domain D). In a large fraction of this thesis we will study constraint language restrictions for various CSP-related problems. (Chapter 6 is in some way an exception—in that chapter we will study constraint language restrictions together with a restriction on the number of occurrences of each variable.) The overreaching goal in these research areas is to classify the complexity of the problem under study for all possible constraint languages. As a first example we will impose constraint language restrictions on CSP. CSP restricted to a constraint language Γ is denoted by CSP(Γ) and is defined as follows.

Definition 1.4 (CSP(Γ))**.** *An instance of* CSP(Γ) *is an instance of the constraint satisfaction problem where each constraint relation is contained in* Γ.

[2] *The International Conference on Principles and Practice of Constraint Programming* is an annual conference on constraint programming. The journal *Constraints* covers constraint programming and related areas.

Note that we have defined an infinite family of problems—there is one problem for each Γ. Two well-known problems, which are CSP(Γ)-problems, are solving systems of linear equations modulo an integer and the 3-satisfiability problem.

Example 1.5 (Linear Equations Modulo 3)**.** *The following problem is a* CSP(Γ) *problem: given a system of linear equations over* $\mathbb{Z}_3$ *(the integers modulo* 3*), is there a solution to the equations?*

Each equation in the system corresponds to one constraint in the CSP instance and each variable corresponds to one variable. The domain of the CSP is $\mathbb{Z}_3 = \{0, 1, 2\}$*. Note that only certain types of constraints are available to us. In particular we can say that* $x + y = 2 \pmod 3$*, but expressing* $x \neq y$ *is not possible. This is a constraint language restriction. As a concrete example of an instance of this problem we can consider the following system of equations:*

$$\begin{aligned} x + y &= 1 \pmod 3 \\ x + z &= 2 \pmod 3 \end{aligned}$$

This instance has three solutions $x = 0, y = 1, z = 2$ *and* $x = 1, y = 0, z = 1$ *and* $x = 2, y = 2, z = 0$*.*

Another example of a CSP(Γ) problem is 3-Satisfiability.

Example 1.6 (3-Satisfiability)**.** *In the* 3-Satisfiability *problem one is given a formula in propositional logic in conjunctive normal form with clauses of length three (3-CNF). That is, a formula on the form*

$$(x_1 \vee x_2 \vee x_3) \wedge (y_1 \vee y_2 \vee y_3) \wedge \dots \tag{1.1}$$

where each $x_1, x_2, x_3, y_1, y_2, y_3$ *is either a positive or negative literal (that is, they are on the form* v *or* $\neg v$ *for some variable* v*).*

It is not hard to see that this problem is equivalent to CSP(Γ) *where* $D = \{0, 1\}$ *and* Γ *consists of all relations which can be expressed as a disjunction with three literals, that is if* $R_{000} = D^3 \setminus (0, 0, 0)$*,* $R_{100} = D^3 \setminus (1, 0, 0)$*,* $R_{110} = D^3 \setminus (1, 1, 0)$*, and* $R_{111} = D^3 \setminus (1, 1, 1)$*, then*

$$\Gamma = \{R_{000}, R_{100}, R_{110}, R_{111}\}.$$

In particular $u \vee v \vee w$ *is equivalent to* $(u, v, w) \in R_{000}$ *and similarly* $\neg u \vee v \vee \neg w$ *is equivalent to* $(u, w, v) \in R_{110}$*. In this way each of the clauses in* (1.1) *can be replaced by a constraint with one of the relations in* Γ*. As a concrete example we can consider the instance*

$$(\neg x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee \neg y \vee \neg z).$$

The equivalent CSP(Γ) *instance is*

$$R_{100}(x, y, z) \wedge R_{110}(y, z, x) \wedge R_{111}(x, y, z).$$

and has five solutions: $(x, y, z) \in \{(0, 0, 0), (0, 0, 1), (0, 1, 0), (1, 0, 1), (1, 1, 0)\}$.

From these two examples one sees that the complexity of CSP(Γ) depends on Γ. It is sometimes solvable in polynomial time, as in Example 1.5, (this problem can be solved in polynomial time by, for example, Gaussian elimination) and sometimes it is **NP**-complete, as in the example above which is equivalent to 3-Satisfiability (3-Satisfiability is probably the most well-known **NP**-complete problem).

The major open problem in this field asks if there is a dichotomy in the complexity of CSP(Γ). That is, is it true that for every constraint language Γ the problem CSP(Γ) is either in **P** or **NP**-complete.[3] This question was first studied by Feder and Vardi [60] and they formulated the following conjecture.

Conjecture 1.7 (The Feder-Vardi Dichotomy Conjecture [60])**.** *For any constraint language* Γ*,* CSP(Γ) *is in* **P** *or is* **NP***-complete.*

As mentioned above this problem is still open, however it has seen considerable progress since it was initially posed. In particular, there is now a conjectured algebraic characterisation of the constraint languages supposedly contained in **P** and those that are **NP**-complete. The algebraic view of CSP(Γ) was first studied in [90] and later refined in [25] where the characterisation was proposed. In Section 11.4 we will describe this characterisation in some more detail.

The algebraic view has seen much study and Conjecture 1.7 is known to hold in a number of special cases. In particular for the following constraint languages the complexity of CSP(Γ) is known:

- two element domains (Schaefer [130]);
- three element domains (Bulatov [18]);
- $\Gamma = \{H\}$ when H is a binary symmetric relation (i.e., an undirected graph) (Hell and Nešetřil [75]);
- $\Gamma = \{H\}$ when H is a directed graph without sources and sinks (Barto et al. [10], note that this is more general than Hell and Nešetřil's result); and
- conservative constraint languages (a constraint language Γ is *conservative* if every unary relation over the domain is contained in Γ) (Bulatov [17, 19]).

In each case it is shown that CSP(Γ) is either in **P** or **NP**-complete, thus giving support to Conjecture 1.7. This list is not exhaustive, however the

[3]Note that for every constraint language Γ, CSP(Γ) is trivially in **NP**. As mentioned in Section 1.1 there is a theorem due to Ladner [110] which states that there are problems of intermediate complexity in **NP** if $\mathbf{P} \neq \mathbf{NP}$. That is, if $\mathbf{P} \neq \mathbf{NP}$, then there are problems in **NP** which are neither in **P** nor **NP**-complete. We can therefore not a priori rule out the possibility that there are CSP(Γ)-problems of intermediate complexity.

results above are easy to describe and they are important pieces in the puzzle of characterising the complexity of CSP(Γ).

In this thesis we will not study the complexity of CSP(Γ) but rather two optimisation problems which are related to CSP(Γ). These two problems are described in Section 1.5 and Section 1.6. We will impose restrictions on these problems in the same way as we imposed restrictions on CSP to define CSP(Γ). The problems we study will thus be parametrised by a constraint language and the complexity of the problem depends on the constraint language.

1.5 Maximum Solution

In Part II of this thesis we will study the complexity of a problem called Max Sol(Γ) which is a variant of CSP(Γ) where we are not only looking for a solution which satisfies all constraints but want a solution which is optimal in a certain sense. For Max Sol we require that the domain D is a subset of the natural numbers $\mathbb{N}$. Formally we define a weighted variant of the problem as follows.

Definition 1.8 (W-Max Sol(Γ))**.**

Instance: *A four-tuple* (V, C, D, w) *where* (V, C, D) *is a* CSP(Γ) *instance and* $w : V \to \mathbb{N}$ *is a* weight function*, and* $D \subset \mathbb{N}$*.*

Solution: *An assignment* $f : V \to D$ *which satisfies all constraints in* C*.*

Measure:

$$\sum_{v \in V} w(v) \cdot f(v)$$

We will sometimes refer to the unweighted variant of the problem above as Max Sol(Γ). In the unweighted variant the weight function assigns the weight 1 to every variable in every instance. The main motivation for studying W-Max Sol is its close connection to integer linear programming: see Examples 1.9 and 1.10 below.

Example 1.9 (Independent Set)**.** *A well-studied problem which is contained in the* Max Sol *framework is the problem of finding a maximum cardinality independent set in a graph. Given a graph* $G = (V, E)$ *a subset* I *of* V *is* independent *if no two vertices in* I *are adjacent. Finding maximum sized independent sets is equivalent to* Max Sol($\{R\}$) *where* R *is the relation* $x + y \leq 1$ *and the domain is* $\{0, 1\}$*. To see this, introduce a variable for every vertex in the graph* G *and add a constraint between any two variables whose corresponding vertices are adjacent to each other in* G*. Conversely, given an instance of* Max Sol($\{R\}$) *we can construct a graph* G *such that there is a one-to-one correspondence between independent sets in* G *and*

solutions to the instance. (And the size of any independent set is equal to the number of ones in the corresponding solution.)

It is known that it is hard to find (quite weak) approximate solutions to Independent Set. *[79, 142] In particular if* $\mathbf{P} \neq \mathbf{NP}$*, then there is no polynomial time algorithm which given a graph* G *produces an independent set* I *of* G *such that* $|I|n^{1-\epsilon} \geq \text{OPT}(G)$*. Here* $\epsilon > 0$ *can be chosen arbitrarily,* n *is the number of vertices in* G*, and* $\text{OPT}(G)$ *is the size of the largest independent set in* G*. Note that it is trivial to find an independent set* I *such that* $|I|n \geq \text{OPT}(G)$*—just pick an arbitrary vertex as the independent set and you are done.*

Example 1.10 (Integer Linear Programming)**.** *In an* integer linear program *(ILP) one is working with linear inequalities over the integers. Given a set of such inequalities we are looking for an integer solution which maximises some linear objective function. The problem can be formulated as: find* $\boldsymbol{x}$ *so that* $\boldsymbol{c}^T\boldsymbol{x}$ *is maximised subject to* $A\boldsymbol{x} \leq \boldsymbol{b}$*,* $\boldsymbol{x}$ *is integral. Here* A*,* $\boldsymbol{c}$*, and* $\boldsymbol{b}$ *are matrices and vectors of appropriate dimensions and* $\boldsymbol{x}$ *is a vector of variables of appropriate dimension.*

If we restrict ourselves to bounded domains of nonnegative integers and nonnegative $\boldsymbol{c}$*, then the ILP problems can straightforwardly be formulated as* Max Sol(Γ) *for a certain* Γ*. Let* $D \subset \mathbb{N}$ *be some fixed finite domain and for each* $n \in \mathbb{N}$ *and each* $\boldsymbol{e} \in \mathbb{R}^n, c \in \mathbb{R}$ *let the relation*

$$\{\boldsymbol{x} \in D^n \mid \boldsymbol{e}^T\boldsymbol{x} \leq c\}$$

be a member of Γ*. It is easy to see that* Max Sol(Γ) *is equivalent to ILP over the domain* D*.*

The thought process lying behind the definition of Max Sol is thus: very many problems can be formulated as ILPs and they have been studied a lot, both from a theoretical point of view and from a practical point of view (see, e.g., [117, 132, 134]). As ILPs are so widely used and studied it would be very interesting and useful to characterise the constraint languages which give rise to tractable Max Sol problems.

The long term goal of the study of Max Sol is to come up with a classification theorem which tells us the complexity Max Sol(Γ) for every constraint language Γ. (Compare with Conjecture 1.7 which is the corresponding conjectured result for CSP.) Here "complexity" can mean different things. One interpretation is a dichotomy result of the type that either Max Sol(Γ) is in $\mathbf{PO}$[4] or it is $\mathbf{NP}$-hard. One can also imagine more fine grained results where the approximability of Max Sol(Γ) is investigated. We obtain such a result in Chapter 5 where it is proved that (for a certain restricted

[4] **PO** is the class of optimisation problems which can be solved in polynomial time. **APX** and **poly-APX** are classes of optimisation problems which can be approximated within a constant factor and a polynomial factor, respectively, in polynomial time. Precise definitions of these classes are given in Chapter 2. We also refer the reader to Chapter 2 for definitions of completeness in these classes.

class of constraint languages) MAX SOL(Γ) is either in **PO**, **APX**-complete, **poly-APX**-complete, it is **NP**-hard to find solution of positive measure, or it is **NP**-hard to find solutions (i.e., CSP(Γ) is **NP**-hard). Note that, as for CSP, it is not known if MAX SOL(Γ) has a dichotomy in its complexity, i.e., is it either in **PO** or **NP**-hard for every Γ? An even more fine grained result could tell us how good approximate solutions one can get by polynomial-time algorithms for MAX SOL(Γ), for every Γ. We establish a result of this type in Chapter 4, for a restricted class of constraint languages.

There are some known results for MAX SOL, which we summarise below.

- When the domain is $\{0,1\}$ MAX SOL is called MAX ONES. For this problem the complexity is known for every constraint language (up to approximability class granularity), see [44, 102].
- When the domain is an interval of integers, i.e., $D = \{a, a+1, a+2, \ldots, b\}$ for some integers $a, b, a \leq b$ and Γ consists of all binary relations of the form $\{(x,y) \in D^2 \mid ax - by \leq c\}$, where a and b are positive integers and c is an arbitrary integer, then MAX SOL(Γ) is solvable in polynomial time. [78]
- When $\Gamma = \{H\}$ and H is a binary symmetric relation (i.e., an undirected graph) there are some partial results [97],
- A *clausal constraint language* consists of relations R where $(\boldsymbol{x}, \boldsymbol{y}) = (x_1, x_2, \ldots, x_p, y_1, y_2, \ldots, y_q) \in R$ if and only if

$$x_1 \geq a_1 \vee x_1 \geq a_2 \vee \ldots \vee x_p \geq a_p \quad \vee$$
$$y_1 \leq b_1 \vee y_2 \leq b_2 \vee \ldots \vee y_q \leq b_q$$

  for some constants $a_1, a_2, \ldots, a_p$ and $b_1, b_2, \ldots, b_q$. In [118] the complexity of MAX SOL was classified for clausal constraint languages (the complexity of CSP for these constraint languages was determined in [43]).

1.6 Maximum CSP

MAX CSP(Γ), which will be studied in Part III, is the variant of CSP(Γ) where we are asked to satisfy as many constraints as possible. Hence, any assignment to the variables is a feasible solution, but some solutions are better than others and we are looking for the best solution. Formally the problem is defined as follows.

Definition 1.11 (W-MAX CSP(Γ))**.**

Instance: *A four-tuple* (V, C, D, w) *where* (V, C, D) *is a* CSP(Γ) *instance,* $w : C \to \mathbb{N}$ *is a* weight function*.*

Solution: *An assignment* $f : V \to D$

Measure:

$$\sum_{c=(\boldsymbol{s},R)\in C} w(c)\cdot R(f(\boldsymbol{s}))$$

In the definition above we have abused the notation slightly, R is used for the indicator function of the relation R. That is, $R(f(\boldsymbol{s}))=1$ if $f(\boldsymbol{s})\in R$ and $R(f(\boldsymbol{s}))=0$ otherwise. Hence, the measure is the sum of the satisfied constraints weighted by w. As for W-Max Sol(Γ) we will sometimes refer to the problem Max CSP(Γ) which is the same as W-Max CSP(Γ) except that the weight function assigns the weight 1 to every constraint in every instance.

Max CSP is a natural optimisation analogue of CSP were we can express preferences among the constraints. If, for example, we are trying to come up with a schedule for the matches in a football tournament it may be desirable that no team plays two matches in two consecutive days. However, if there are no schedules with this property, then we are better off with a schedule which minimises such pair of matches compared to no schedule at all. Note that we would get the latter result if the problem was simply treated as an ordinary CSP.

The optimisation problems mentioned in Section 1.1 can in fact be seen as Max CSP(Γ) problems. As an example we will have a look at Minimum Cut.

Example 1.12 (Minimum Cut as a W-Max CSP)**.** *In* Minimum Cut *one is given a graph $G=(V,E)$ and two vertices s and v and is asked to find a* cut*, that is a partition of $V\setminus\{s,t\}$ into two sets C and C', of minimum size. The* size *of a cut (C,C') is the number of edges with one vertex in $C\cup\{s\}$ and the other in $C'\cup\{t\}$. It is known that a minimum cut can be found in time polynomial in $|V|$.*

To model Minimum Cut *as a* W-Max CSP(Γ) *we let $D=\{0,1\}$ and $\Gamma=\{c_0,c_1,EQ_D\}$. Here c_0 and c_1 are the unary relations $\{(0)\}$ and $\{(1)\}$, respectively. EQ_D is the equality relation on D, so $EQ_D=\{(0,0),(1,1)\}$. Given a graph $G=(V,E)$ we construct an instance $I=(V,C,D,w)$ of* W-Max CSP(Γ) *by adding a constraint $((v,v'),EQ_D)$ for every pair of vertices v,v' in V which are adjacent in G. Finally, we add the constraints (s,c_0) and (t,c_1) to C. The weight function w is defined by $w((s,c_0))=w((t,c_1))=|E|+1$ and $w(\boldsymbol{c})=1$ for all other constraints $\boldsymbol{c}$. Note that in any optimal solution f to I we will have $f(s)=0$ and $f(t)=1$, due to the large weights on (s,c_0) and (t,c_1). Furthermore, as f is optimal, the partition*

$$(\{v\in V\setminus\{s\}\mid f(v)=0\},\{v\in V\setminus\{t\}\mid f(v)=1\})$$

will be a cut of minimum size in G. It is known that W-Max CSP(Γ) *can be solved in polynomial time. [102] The algorithm does in fact consist of a reduction to* Minimum Cut*.*

If we instead want to find a cut in a graph with the *maximum* number of edges we arrive at the problem Max Cut. This problem is equivalent to Max CSP($\{(0,1),(1,0)\}$), that is, the domain contains two values, 0 and 1, and we have one relation, namely, the inequality relation. It is known that finding optimal solutions to Max Cut is **NP**-hard [66]. The (in)approximability of Max Cut is also well-studied, see, e.g., [68, 81, 104].

By our observations above there are constraint languages which give rise to tractable problems for Max CSP and other constraint languages which give rise to non-tractable problems. As for Max Sol we will be interested in classifying the constraint languages according to their (non-)tractability for Max CSP. A considerable amount of research effort has been spent on Max CSP and there is now a conjectured classification of the constraint languages which are tractable. Very strong conditional approximability results with matching inapproximability results are also known for many constraint languages for Max CSP. In Part III of this thesis we will study the complexity of Max CSP and get some new results for this problem. The first chapter of Part III, Chapter 7, contains a much more thorough introduction to Max CSP. We refer the reader to that chapter for details regarding the conjectured classification and the work that has been done on Max CSP, including references to the relevant papers.

1.7 Main Contributions

This section contains a summary of the main contributions of this thesis. As we have not yet introduced all the concepts needed to state the results formally, we refer the reader to the corresponding parts of the thesis for the full details.

In Part II various variants of Max Sol are studied. The main contributions in this part of the thesis are as follows:

- In Chapter 4 the approximability of Max Sol(Γ) is studied when Γ contains all relations which can be seen as equations over some finite abelian group. The main result is a tight approximability result for Max Sol(Γ). That is, for every abelian group G it is shown that there is some number α such that solutions to Max Sol(Γ) can be found in polynomial time which are at most an α-fraction away from the optimum solution. Furthermore, it is shown that finding solutions which are only an $(\alpha-\epsilon)$-fraction away from the optimum is **NP**-hard, for any $\epsilon > 0$.

- In Chapter 5 so-called *maximal constraint languages* are investigated for Max Sol. These languages have received some attention for other similar problems such as CSP [26, 20] and quantified CSP [31].

  We show a classification theorem for these constraint languages for domains of size four or less. We also give a complete classification theorem under the assumption of a certain conjecture.

- In Chapter 6 Max Sol is studied in the boolean domain with the additional restriction that each variable only occurs a bounded number of times. When the bound is greater than two we give a complete characterisation theorem. When the bound is two we give some partial results.

In Part III a number of results regarding the complexity of Max CSP are obtained. The *submodular function minimisation* (SFM) problem is intimately connected to the complexity of Max CSP. In particular, all known tractable classes of Max CSP are reductions to some SFM problem. In an SFM problem one wants to minimise a certain function defined over a product of a special kind of partial orders called lattices. We refer the reader to Chapter 7 for further details regarding this problem and its connection to Max CSP. The main contributions in Part III are:

- In Chapter 8 we show that the SFM problem can be solved in pseudopolynomial time over a product of *diamonds*. Diamonds are a special kind of lattices for which no complexity results for SFM were known before this work.

- In Chapter 9 we show that the SFM problem over a product of modular lattices is contained in an appropriately modified variant of **coNP**, thus placing the problem in $\mathbf{NP} \cap \mathbf{coNP}$ (again appropriately modified). This gives evidence in support of the hypothesis that this class of problems can be solved efficiently.

- In Chapter 10 some observations are made of what can be achieved for SFM by combining existing known techniques. In particular, it is shown that the classes of lattices for which SFM is tractable (for various definitions of "tractable") are closed under taking sublattices.

- In Chapter 11 it is shown that constraint languages Γ which are known to make CSP(Γ) **NP**-complete makes Max CSP(Γ) hard to approximate in the sense that one cannot obtain arbitrarily good approximate solutions in polynomial time, unless $\mathbf{P} = \mathbf{NP}$.

- In Chapter 12 single relation Max CSP is studied. We show that the complexity of Max CSP($\{R\}$) is either trivial or **NP**-hard to approximate arbitrarily well.

1.8 Organisation

This thesis is structured as follows: this chapter, which you are reading now, contains some background material and sets the general scene for what will come in the subsequent parts. In Chapter 2 some technical preliminaries are given which will be used throughout the thesis. In some chapters we need additional definitions and these will be introduced when the need arise. In

the final chapter of the first part we introduce the *valued constraint satisfaction problem* (VCSP) and discuss some connections between CSP, Max CSP, Max Sol, and VCSP.

Part II contains three chapters which all concern the complexity of variants of Max Sol. In the first chapter, number 4, Max Sol is studied when we have constraints that are equations over finite groups. In Chapter 5 the complexity of *maximal* constraint languages (see Chapter 5 for a definition) for Max Sol is studied. Finally, in the last chapter of Part II, we investigate a variant of Max Sol over the boolean domain where each variable occurs a bounded number of times.

Part III is devoted to the complexity of Max CSP. Chapter 7 contains an introduction to Max CSP. The subsequent chapters of Part III can be divided into two subparts. In the first subpart, which contains chapters 8, 9, and 10, certain tractability results are obtained for the submodular function minimisation problem, which is a problem intimately connected to Max CSP. The other subpart consists of Chapter 11 and Chapter 12. In Chapter 11 constraint languages which are known to be hard for CSP are investigated in the Max CSP setting. In the last chapter in Part III, Chapter 12, we take a look at the constraint languages which consists of a single relation.

Finally, Part IV contains a short discussion of possible future work.

Chapter 2

Technical Preliminaries

In this chapter we will state some definitions which will be used throughout the thesis. In some chapters we need some additional definitions, these will be introduced when the need arise. As mentioned in Section 1.2 R_D is the set of all finitary relations over the domain D. Given a relation $R \in D$ and tuple $\boldsymbol{x} \in D^n$ we will sometimes write $R(\boldsymbol{x})$ with the meaning $\boldsymbol{x} \in R$. Furthermore, we will sometimes use R as the indicator function for the relation R, i.e., $R(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in R$ and $R(\boldsymbol{x}) = 0$ otherwise. It will be clear from the context which of these interpretations one should assume to be in use.

For a subset $X \subseteq D$ and an n-ary relation $R \in R_D$ we let the *restriction of R onto X*, denoted by $R|_X$, be the relation

$$\{\boldsymbol{x} \in R \mid \boldsymbol{x} \in X^n\}.$$

We extend this to sets of relations $\Gamma \subseteq R_D$ in the usual way: $\Gamma|_X = \{R|_X \mid R \in \Gamma\}$. We will use a similar notation for functions; if D, X, and n are as above and f is a function from D^n to D we use $f|_X$ to denote the function $g : X^n \to D$ defined by $g(\boldsymbol{x}) = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in X^n$. For a set of functions F on D we let $F|_X = \{f|_X \mid f \in F\}$.

Given an n-ary relation $R \in R_D$, a positive integer m, and a subset $I = \{i_1, i_2, \ldots, i_m\} \subseteq [n]$ where $i_1 < i_2 < \ldots < i_m$ the *projection of R onto I*, denoted by $\mathrm{pr}_I\, R$, is the m-ary relation

$$\{(\boldsymbol{x}(i_1), \boldsymbol{x}(i_2), \ldots, \boldsymbol{x}(i_m)) \mid \boldsymbol{x} \in R\}.$$

Recall that a *digraph* is a pair (V, E) such that V is a finite set and $E \subseteq V \times V$. A *graph* is a digraph (V, E) such that for every pair $(x, y) \in E$ we also have $(y, x) \in E$.

Given a set X and a function $f : X \to \mathbb{R}$ we use $\arg\max_{x \in X} f(x)$ to denote the maximisers of f, i.e., the set $\{x' \in X \mid f(x') = \max\{f(x) \mid x \in X\}\}$.

2.1 Approximability, Reductions, and Completeness

A *combinatorial optimisation problem* is defined over a set of *instances* (admissible input data) $\mathcal{I}$; each instance $I \in \mathcal{I}$ has a finite set $\text{SOL}(I)$ of *feasible solutions* associated with it. The objective is, given an instance I, to find a feasible solution of *optimum* value with respect to some measure function m defined for pairs (x, y) such that $x \in \mathcal{I}$ and $y \in \text{SOL}(x)$. Every such pair is mapped to a nonnegative integer by m. The optimal value is the largest one for *maximisation* problems and the smallest one for *minimisation* problems. A combinatorial optimisation problem is said to be an **NPO** problem if its instances and solutions can be recognised in polynomial time, the size of the solutions are polynomially-bounded in the size of the input, and the objective function can be computed in polynomial time (see, e.g., [9]).

Definition 2.1 (r-approximate)**.** *A solution $s \in \text{SOL}(I)$ to an instance I of an* **NPO** *problem Π is* r-approximate *if*

$$\max\left\{\frac{m(I,s)}{\text{OPT}(I)}, \frac{\text{OPT}(I)}{m(I,s)}\right\} \leq r,$$

where $\text{OPT}(I)$ is the optimal value for a solution to I.

An approximation algorithm for an **NPO** problem Π has *performance ratio* $\mathcal{R}(n)$ if, given any instance I of Π with $|I| = n$, it outputs an $\mathcal{R}(n)$-approximate solution. (We use $|I|$ to denote the size of an encoding of I.)

Definition 2.2 (**PO**, **APX**, **poly-APX**)**. PO** *is the class of* **NPO** *problems that can be solved (to optimality) in polynomial time. An* **NPO** *problem Π is in the class* **APX** *if there is a polynomial-time approximation algorithm for Π whose performance ratio is bounded by a constant. Similarly, Π is in the class* **poly-APX** *if there is a polynomial-time approximation algorithm for Π whose performance ratio is bounded by a polynomial in the size of the input.*

A *polynomial-time approximation scheme* (PTAS) is an approximation algorithm which for an arbitrary $\epsilon > 0$ produces a solution with performance ratio $1+\epsilon$ and the time required by the algorithm is bounded by a polynomial in n for each fixed ϵ (here n denotes the size of the instance). Running times such as $O(n^{1/\epsilon})$ and even $O\left(n^{f(1/\epsilon)}\right)$ for any function f are thus OK for a PTAS. Note that for each fixed ϵ the running time of the algorithm is polynomial in the size of the instance. That an optimisation problem admits a PTAS is a highly desirable property, which many problems do not have under reasonable complexity theoretic conjectures (such as $\mathbf{P} \neq \mathbf{NP}$).

To relate the difficulty of approximating different optimisation problems we will use reductions. There are several different types of approximation preserving reductions. We will make use of AP-reductions, L-reductions, and S-reductions. The AP-reduction defines hardness and completeness for **APX** and **poly-APX**, but it is sometimes a bit difficult to

use. The L-reductions sometimes implies the existence of an AP-reduction (see Lemma 2.5 below for the exact conditions) and are a bit easier to use than the AP-reductions. S-reductions are simplest of them all, but they are often not applicable. We begin with defining AP-reductions. The definition below follows the definition in [44, 102].

Definition 2.3 (AP-reduction [44, 102])**.** *Given two* **NPO** *problems* Π_1 *and* Π_2 *an* AP-reduction *from* Π_1 *to* Π_2 *is a triple* (F, G, α) *such that,*

- F *and* G *are polynomial-time computable functions and* $\alpha > 0$ *is a constant;*
- *for any instance* I *of* Π_1, $F(I)$ *is an instance of* Π_2*;*
- *for any instance* I *of* Π_1, *and any feasible solution* s' *of* $F(I)$, $G(I, s')$ *is a feasible solution of* I*;*
- *for any instance* I *of* Π_1, *and any* $r \geq 1$, *if* s' *is an* r*-approximate solution of* $F(I)$ *then* $G(I, s')$ *is an* $(1 + (r-1)\alpha + o(1))$*-approximate solution of* I *where the o-notation is with respect to* $|I|$.

If such a triple exist we say that Π_1 is AP-reducible to Π_2. *We use the notation* $\Pi_1 \leq_{AP} \Pi_2$ *to denote this fact.*

It is a well-known fact (see, e.g., Section 8.2.1 in [9]) that AP-reductions compose. As mentioned above we will sometimes use L- and S-reductions instead of AP-reductions as they are easier to work with. L-reductions are defined as follows.

Definition 2.4 (L-reduction [9])**.** *Given two* **NPO** *problems* Π_1 *and* Π_2 *an* L-reduction *from* Π_1 *to* Π_2 *is a quadruple* (F, G, β, γ) *such that,*

- F *and* G *are polynomial-time computable functions and* $\beta > 0$ *and* $\gamma > 0$ *are constants;*
- *for any instance* I *of* Π_1, $F(I)$ *is an instance of* Π_2, *such that*

$$\text{OPT}(F(I)) \leq \beta \cdot \text{OPT}(I);$$

- *given* I, *and any solution* s' *to* $F(I)$, *the algorithm* G *produces a feasible solution* $s = G(I, s')$ *to* I *such that*

$$|\text{OPT}(I) - m_1(I, s)| \leq \gamma \cdot |\text{OPT}(F(I)) - m_2(F(I), s')|,$$

where m_1 *is the measure for* Π_1 *and* m_2 *is the measure for* Π_2.

If such a quadruple exist we say that Π_1 is L-reducible to Π_2. *We use the notation* $\Pi_1 \leq_L \Pi_2$ *to denote this fact.*

It is known (see, e.g., Lemma 8.2 in [9]) that, if Π_1 is L-reducible to Π_2 and $\Pi_1 \in$ **APX** then there is an AP-reduction from Π_1 to Π_2. We state this as a lemma.

Lemma 2.5. *If* $\Pi_1 \leq_L \Pi_2$ *and* $\Pi_1 \in$ **APX**, *then* $\Pi_1 \leq_{AP} \Pi_2$.

S-reductions are similar to *L*-reductions but instead of the condition $\text{OPT}(F(I)) \leq \beta \cdot \text{OPT}(I)$ we require that $\text{OPT}(F(I)) = \text{OPT}(I)$ and instead of $|m_1(I,s) - \text{OPT}(I)| \leq \gamma \cdot |m_2(F(I),s') - \text{OPT}(F(I))|$ we require that $m_1(I,s) = m_2(F(I),s')$. [46] If there is an *S*-reduction from Π_1 to Π_2 (written as $\Pi_1 \leq_S \Pi_2$), then there is an *AP*-reduction from Π_1 to Π_2. An **NPO** problem Π is **APX**-*hard* (**poly-APX**-*hard*) if every problem in **APX** (**poly-APX**) is *AP*-reducible to it. If, in addition, Π is in **APX** (**poly-APX**), then Π is **APX**-*complete* (**poly-APX**-*complete*).

2.2 Co-clones and Polymorphisms

In the CSP setting one can sometimes prove that $\text{CSP}(\Gamma \cup \{R\})$ is no harder than $\text{CSP}(\Gamma)$. That is, for suitable R and Γ $\text{CSP}(\Gamma \cup \{R\})$ is reducible to $\text{CSP}(\Gamma)$ in polynomial time. In particular, if $R_1, R_2, \ldots, R_n \in \Gamma$ and R can be defined as

$$R(\boldsymbol{x}) \iff \exists \boldsymbol{y} : R_1(\boldsymbol{z_1}) \wedge \ldots \wedge R_n(\boldsymbol{z_n}) \tag{2.1}$$

(here $\boldsymbol{z_1}, \ldots, \boldsymbol{z_n}$ contains variables from $\boldsymbol{x}$ and $\boldsymbol{y}$), then $\text{CSP}(\Gamma \cup \{R\})$ is no harder than $\text{CSP}(\Gamma)$. To see this, note that given an instance I of $\text{CSP}(\Gamma \cup \{R\})$ we can replace any constraint application which has R as its constraint relation by the formula (2.1) (we introduce fresh variables for $\boldsymbol{y}$ in each replacement). It is not hard to see that the new instance, I', is an instance of $\text{CSP}(\Gamma)$ and that I has a solution if and only if I' has a solution. The procedure just described is a polynomial-time many-one reduction from $\text{CSP}(\Gamma \cup \{R\})$ to $\text{CSP}(\Gamma)$.

This phenomena (that certain relations can be added to the constraint language without increasing the complexity of the problem) has been studied extensively. [25, 90] In this section we will briefly describe an algebraic theory which tells us something about the set of relations that can be defined by a constraint language in this way.

A relation R is *pp-definable* (primitive positive definable) in Γ if $\boldsymbol{x} \in R$ is equivalent to a first order formula consisting of relations in $\Gamma \cup \{EQ_D\}$, conjunction and existential quantification (here EQ_D is the equality relation on the domain D). So a pp-definition is of the form (2.1) except that equality constraints are allowed as well.

Example 2.6 (pp-definition)**.** *Let the domain* D *be* $\{1,2,3\}$ *and let the constraint language* Γ *be* $\{c_0, <\}$ *where* $c_0 = \{(0)\}$ *and* $<$ *denotes the usual strictly-less-than relation restricted to the domain* D*. We can then pp-define the unary relation* $R = \{1,2\}$ *as*

$$R(x) \iff \exists y : c_0(y) \wedge y < x.$$

We define a closure operator, $\langle\cdot\rangle$, on constraint languages as follows: for a constraint language Γ the set $\langle\Gamma\rangle$ consists of all relations that can be pp-defined in Γ.

An *operation* is a function f from D^k to D for some positive integer k. Any operation can be extended to a function over n-tuples as follows: let $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}$ be k tuples in D^n then $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k})$ is defined to be the tuple

$$\big(f(\boldsymbol{t_1}(1), \boldsymbol{t_2}(1), \ldots, \boldsymbol{t_k}(1)), \ldots, f(\boldsymbol{t_1}(n), \boldsymbol{t_2}(n), \ldots, \boldsymbol{t_k}(n))\big).$$

Given a n-ary relation R we say that R is *invariant* (or, closed) under f if

$$\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k} \in R \Rightarrow f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) \in R.$$

Conversely, for an operation f and a relation R, f is a *polymorphism* of R if R is invariant under f. For a constraint language Γ we say that Γ is invariant under f if every relation in Γ is invariant under f. We analogously extend the notion of polymorphisms to constraint languages, i.e., an operation f is a polymorphism of Γ if Γ is invariant under f. Those concepts have been very useful in the study of the complexity of various constraint satisfaction problems (see, e.g., [17, 22, 25, 90]).

Example 2.7. *Let $D = \{0, 1, 2\}$ and let R be the directed cycle on D, i.e., $R = \{(0, 1), (1, 2), (2, 0)\}$. One polymorphism of R is the operation $f : \{0, 1, 2\}^3 \to \{0, 1, 2\}$ defined as $f(x, y, z) = x - y + z \pmod 3$. This can be verified by considering all possible combinations of three tuples from R and evaluating f component-wise.*

The set of polymorphisms for a constraint language Γ will be denoted by $\mathrm{Pol}(\Gamma)$, and for a set of operations C the set of all relations which are invariant under C will be denoted by $\mathrm{Inv}(C)$.

Given some domain D and positive integer n an operation $f : D^n \to D$ is a *projection* if $f(x_1, x_2, \ldots, x_n) = x_k$ for some $k \in [n]$. A *clone* is a set of operations, which is closed under composition and contains all projections (so, if f is an n-ary operation in some clone C and $g_1, g_2, \ldots, g_n$ are m-ary operations in C, then the m-ary operation $f(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_n(\boldsymbol{x}))$ is contained in C).

The sets $\mathrm{Pol}(\Gamma)$ are clones. For a set of operations C, $\mathrm{Inv}(C)$ is called a *relational clone* or a *co-clone*. It is well-known that the co-clones (and the clones) form a lattice (that is, a partial order with least upper bounds and greatest lower bounds) under the subset order. Over the boolean domain Emil Post has classified all co-clones and their inclusion structure [125]. Over larger domains the structure of the co-clones (and clones) is much more complicated and the lattice is largely unknown. We will use a part of Post's classification in Chapter 6 (the part we use is diagrammed in Figure 6.1).

The following theorem relates the pp-definable relations for a constraint language to the set of polymorphisms of the language.

Theorem 2.8 ([126])**.**

- *For every set* $\Gamma \subseteq R_D$, $\langle\Gamma\rangle = \mathrm{Inv}(\mathrm{Pol}(\Gamma))$.
- *If* $\Gamma' \subseteq \langle\Gamma\rangle$, *then* $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\Gamma')$.

There is a considerable body of results which relates the existence of certain nice polymorphisms to the tractability of the corresponding CSP. One fairly early example is the following: a binary function f on D is a *semilattice operation* if it is idempotent, commutative and associative.

Theorem 2.9 ([90])**.** *If* $\Gamma \subseteq R_D$ *and there is a semilattice operation in* $\mathrm{Pol}(\Gamma)$, *then* $\mathrm{CSP}(\Gamma)$ *is tractable.*

Similar theorems in the same style now exists for a variety of different polymorphisms, see, e.g., [17, 21, 23].

Chapter 3

Connections to VCSP

The Valued Constraint Satisfaction Problem (VCSP) is a very general framework which contains two of the problems defined in Chapter 1, CSP and Max CSP, as special cases. VCSP also contains a variant of W-Max Sol as a special case. We will define VCSP in this chapter and describe the connections to CSP, Max CSP, and Max Sol. (As far as we know VCSP was first introduced in [131] by Schiex et al. However, the variant defined in [131] is even more general than the definition we use here.)

Instead of working with sets of relations or predicates the problem is parametrised by a set of cost functions. A *cost function* is a function from D^k, for some positive integer k, to $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{-\infty\}$. Here $-\infty$ should be seen as a special number which is less than every number in $\mathbb{Q}$ and satisfies $x + -\infty = -\infty$ for all $x \in \overline{\mathbb{Q}}$. (We will never use any other operations on numbers from $\overline{\mathbb{Q}}$ besides addition.) A set of cost functions is a *valued constraint language*.

Definition 3.1 (Valued Constraint Satisfaction Problem)**.** *The valued constraint satisfaction problem over the valued constraint language* Γ *is denoted by* VCSP(Γ) *and is the maximisation*[1] *problem with*

Instance: *A tuple* (V, C, D) *where*

- V *is a set of variables;*
- D *is a set of values (the* domain*); and*
- C *is a multiset of constraints* $\{C_1, \ldots, C_q\}$*, in which each constraint* C_i *is a pair* $(\boldsymbol{s_i}, g_i)$*, where* $\boldsymbol{s_i}$ *is a list of variables of length* m_i*, called the* constraint scope*, and* $g_i \in \Gamma$ *is an* m_i*-ary cost function.*

[1]The VCSP is often defined to be a minimisation problem instead of a maximisation problem. In the minimisation case the cost functions map tuples of the domain to $\mathbb{Q} \cup \{\infty\}$ instead of $\mathbb{Q} \cup \{-\infty\}$ as we do here. There are no conceptual differences between the two definitions, we use the maximisation variant here because it is consistent with how we have defined Max CSP and Max Sol.

Solution: *An assignment* $f : V \to D$

Measure:

$$\sum_{(\boldsymbol{s},g)\in C} g(f(\boldsymbol{s}))$$

Although not as intensively studied as CSP, VCSP has received considerable attention from various researchers. Some of the known results are:

- a complete characterisation of the complexity of VCSP for the boolean domain [36];
- an algebraic characterisation of the expressibility of valued constraint languages [33] (this characterisation is reminiscent of the well-known algebraic characterisation of the expressibility of ordinary constraint languages presented in Section 2.2);
- faster algorithms for specific valued constraint languages [39];
- additional tractable cases [34]; and
- tight approximability and inapproximability results for a large class of valued constraint languages under the unique games conjecture [127] (this result will be described in a little more detail in Chapter 7).

(In some of the results above the range of the cost functions are restricted to the nonnegative rational numbers and $-\infty$.) In the VCSP framework different tuples may have different values according to the cost function. One can compare this with the CSP case where every tuple is either contained in a certain relation or not contained in it and to the Max CSP case where each tuple is either contained in the relation in which case this tuple contributes 1 to the measure of the solution, or not contained in the relation in which case it does not give any contribution to the measure.

We will now sketch the arguments needed to show that VCSP contains CSP, Max CSP and a variant of Max Sol as special cases. More specifically, we want to show that for every domain D and every constraint language Γ over D there are valued constraint languages Γ_{CSP}, $\Gamma_{\text{Max CSP}}$ and $\Gamma_{\text{Max Sol}}$ such that $\mathrm{CSP}(\Gamma)$ is polynomially time equivalent to $\mathrm{VCSP}(\Gamma_{\mathrm{CSP}})$, Max $\mathrm{CSP}(\Gamma)$ is polynomially time equivalent to $\mathrm{VCSP}(\Gamma_{\text{Max CSP}})$ and finally that the variant of Max $\mathrm{Sol}(\Gamma)$ is polynomially time equivalent to $\mathrm{VCSP}(\Gamma_{\text{Max Sol}})$.

To this end let Γ be an arbitrary constraint language. We can construct a valued constraint language Γ_{CSP} from Γ by introducing, for each k-ary relation $R \in \Gamma$, a k-ary cost function f in Γ_{CSP} such that $f(\boldsymbol{x}) = 0$ if $\boldsymbol{x} \in R$ and $f(\boldsymbol{x}) = -\infty$ otherwise. It is not hard to see that $\mathrm{CSP}(\Gamma)$ and $\mathrm{VCSP}(\Gamma_{\mathrm{CSP}})$ are equivalent problems. In particular, for every instance I of $\mathrm{CSP}(\Gamma)$ and every assignment s to V we can see $I' = I$ as an instance of $\mathrm{VCSP}(\Gamma_{\mathrm{CSP}})$, furthermore s is a solution to I if and only if the measure of s on I' is not $-\infty$.

From Γ we can construct another valued constraint language, $\Gamma_{\text{Max CSP}}$, such that for every k-ary relation $R \in \Gamma$ we introduce a k-ary cost function f in $\Gamma_{\text{Max CSP}}$ such that $f(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in R$ and $f(\boldsymbol{x}) = 0$ otherwise. In this case Max CSP(Γ) and VCSP($\Gamma_{\text{Max CSP}}$) are equivalent problems.

If we assume that $D \subset \mathbb{N}$, then we can construct yet another valued constraint language, $\Gamma_{\text{Max Sol}}$, from Γ as follows. Let i be the identity function on D, that is, $i(x) = x$ for all $x \in D$. Now let $\Gamma_{\text{Max Sol}} = \Gamma_{\text{CSP}} \cup \{i\}$. With these choices of Γ and $\Gamma_{\text{Max Sol}}$ there is a certain relation between (W-)Max Sol(Γ) and VCSP($\Gamma_{\text{Max Sol}}$). In particular, for any instance $I = (V, D, C)$ of Max Sol(Γ) we can construct an instance $I' = (V, D, C')$ of VCSP($\Gamma_{\text{Max Sol}}$) simply by adding the unary constraint $((v), i)$ to C' for every variable v (so $C' = C \cup \{((v), i) \mid v \in V\}$). For an assignment s to V, if s is a feasible solution to I with measure m, then s is a solution to I' with measure m. If s is not a feasible solution to I, then the measure of s on I' is $-\infty$. The converse does also hold: assume that s has measure m on I', then s has measure m on I if $m \neq -\infty$, and otherwise, if $m = -\infty$, then s is not a feasible solution to I.

To show that the two problems are equivalent we also need to construct an instance I of Max Sol(Γ) for every instance I' of VCSP($\Gamma_{\text{Max Sol}}$). However, there is a problem here because there might be some variable v for which there are two constraints of the form $((v), i)$. We do not know how to model this in Max Sol(Γ), but if we allow ourselves to use weights (that is, we construct an instance of W-Max Sol(Γ) instead), then we can get around this obstacle by letting the weight of v be the number of occurrences of the constraint $((v), i)$ in I'. However, we cannot construct equivalent instances of VCSP($\Gamma_{\text{Max Sol}}$) from instances of W-Max Sol(Γ) with the approach used above as the value of a weight w is exponential in its encoding length $O(\log w)$. By using these constructions one can prove that VCSP($\Gamma_{\text{Max Sol}}$) is polynomially time equivalent to W-Max Sol(Γ) with the additional restriction that the weights are bounded by a polynomial in the size of the instances (so for any instance $I = (V, D, C, w)$ we have $w(v) \leq (|V| + |D| + |C|)^c$ for some fixed constant c). Another approach is to introduce weights in the definition of VCSP and show that weighted VCSP($\Gamma_{\text{Max Sol}}$) is equivalent to W-Max Sol(Γ).

The conclusion of these observations is that by studying CSP(Γ), Max CSP(Γ), and Max Sol(Γ) one essentially studies VCSP for certain restricted classes of valued constraint languages. Classification and dichotomy results are, of course, easier to come by if one restricts the class of (valued) constraint languages under study. This can also be observed in our current knowledge: there are, currently, conjectured classification theorems for CSP and Max CSP, but no such thing for Max Sol or VCSP. Additionally, there are a number of known classification theorem for restricted classes of constraint languages for CSP and Max CSP, see Section 1.4 and Chapter 7. There are a few such results for Max Sol and VCSP (maybe the most notable ones are the classification theorems for the two element domains), but CSP and Max CSP are certainly better understood.

Part II

Maximum Solution

Chapter 4

Max Sol over Abelian Groups

4.1 Introduction

Let G be a finite abelian group with identity element 0_G. In this chapter we are going to study the following problem.

Definition 4.1 (W-Max Sol Eqn(G, g))**.** Weighted Maximum Solution Equation(G, g) *where, G is a group and $g : G \to \mathbb{N}$ is a function, is denoted by* W-Max Sol Eqn(G, g)*. An instance of* W-Max Sol Eqn(G, g) *is defined to be a triple (V, E, w) where,*

- *V is a set of variables;*
- *E is a set of equations of the form $w_1 + \ldots + w_k = 0_G$, where each w_i is either a variable, an inverted variable or a group constant;*
- *$w : V \to \mathbb{N}$ is a weight function.*

The objective is to find an assignment $f : V \to G$ to the variables such that all equations are satisfied and the sum

$$\sum_{v \in V} w(v)g(f(v))$$

is maximised.

Note that the function g and the group G are not parts of the input. Thus, W-Max Sol Eqn(G, g) is a problem parametrised by both the group G and the function g. This is a bit different from the Max Sol problem where we only parametrise on the constraint language. If g is injective, then W-Max Sol Eqn(G, g) can be seen as a W-Max Sol problem. In this

case the constraint language in the corresponding W-MAX SOL problem is given by G and g. On the other hand, if g is not injective then there is no straightforward translation of W-MAX SOL EQN(G, g) to W-MAX SOL(Γ). However, regardless of the properties of g, MAX SOL EQN(G, g) can always be seen as a VCSP problem. We will use some of the results obtained in this chapter in Chapter 5 when we study the maximal constraint languages for MAX SOL.

One could also consider studying W-MAX SOL EQN for non-abelian groups, but due to a result of Goldmann and Russell [69] it is **NP**-hard to find feasible solutions to this problem. We do therefore restrict ourselves to abelian groups. As G is abelian, the collection of linear equations in an instance of W-MAX SOL EQN$(G,\ g)$ can be represented in the standard way as an integer-valued matrix, A, and a vector of group elements, $\boldsymbol{b}$. If the variables are called $x_1, \ldots, x_m$ we can then, with $\boldsymbol{x} = (x_1, \ldots, x_m)^T$, use $A\boldsymbol{x} = \boldsymbol{b}$ as an equivalent form of the sets V and E in the definition above. To describe our results we need a couple of definitions.

For a group G and a subgroup $G' \subseteq G$ of this group we denote the coset, C, of G' with representative $c \in G$ as $G' + c$. That is,

$$G' + c = C = \{x + c \mid x \in G'\}.$$

We note that given A, $\boldsymbol{b}$, and $\boldsymbol{x}$ as above, the set of solutions to $A\boldsymbol{x} = \boldsymbol{b}$ is a coset of G^n (if the set is non-empty). To see this note that if $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ are solutions to the system of equations, then $\boldsymbol{x} - \boldsymbol{y} + \boldsymbol{z}$ is a solution as well. This implies that the set of solutions is a coset.

For a function $f : X \to \mathbb{N}$ and a set $S \subseteq X$ we use the notations $f_{\max}(S)$ and $f_{\text{sum}}(S)$ for the quantities,

$$\max_{x \in S} f(x) \quad \text{and} \quad \sum_{x \in S} f(x),$$

respectively. We will sometimes use $f_{\max}$ and f_{sum} instead of $f_{\max}(X)$ and $f_{\text{sum}}(X)$, respectively. Those notations will only be used when they are well defined. We also need the following definition.

Definition 4.2 (Equation Coset)**.** *Let G be an abelian group. A coset $B \subseteq G$ is an* equation coset *if there exists a matrix A, a vector $\boldsymbol{b}$, and a vector of variables $\boldsymbol{x} = (x_1, \ldots, x_m)^T$ such that the system of equations $A\boldsymbol{x} = \boldsymbol{b}$ restrict the values that x_1 can have to B. That is,*

$$B = \{x_1 \mid A\boldsymbol{x} = \boldsymbol{b}\}$$

for some matrix A and vector $\boldsymbol{b}$.

Given a group G there is always at least one coset that is an equation coset, namely G itself. Not all cosets of an abelian group are necessarily equation cosets. As an example let $G = \mathbb{Z}_2 \times \mathbb{Z}_2$ and consider the coset $C = \{(0,0),(1,1)\}$. It is easy to see that C is a coset (it is in fact a

subgroup), but it is not an equation coset. To see this, note that we can perform Gaussian elimination on A and $\boldsymbol{b}$. Furthermore, as $G = \mathbb{Z}_2 \times \mathbb{Z}_2$ we can, after each step in the Gaussian elimination process, reduce the entries of A modulo 2 without changing the set of solutions. At the end we will end up with an equation of the form $x_1 = x_{i_1} + x_{i_2} + \ldots + x_{i_k} + c$ for some integers $k, i_1, i_2, \ldots, i_k$ and group element c. Furthermore, $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ are free parameters in the sense that any assignment to these variables will yield a solution. It follows that the set of values x_1 takes in the solutions is either a single group element or the entire group, in particular x_1 cannot only take the values $(0,0)$ and $(1,1)$.

For a group G we write Eqn-Coset(G) to denote the collection of all $B \subseteq G$ such that B is an equation coset. The main result of this chapter is the following theorem about the approximability of W-Max Sol Eqn(G, g).

Theorem 4.3. *For every finite abelian group G and every function $g : G \to \mathbb{N}$,* W-Max Sol Eqn(G, g) *is approximable within α where*

$$\alpha = \max\left\{ |B| \cdot \frac{g_{\max}(B)}{g_{\mathrm{sum}}(B)} \;\middle|\; B \in \text{Eqn-Coset}(G) \right\}.$$

Furthermore, for every finite abelian group G and every nonconstant function $g : G \to \mathbb{N}$ W-Max Sol Eqn(G, g) *is not approximable within $\alpha - \epsilon$ for any $\epsilon > 0$ unless* $\mathbf{P} = \mathbf{NP}$.

We will prove the inapproximability part of Theorem 4.3 in Section 4.2 (Theorem 4.4) and the approximability part in Section 4.3 (Theorem 4.12). In Section 4.2.4 we also show that W-Max Sol($\mathbb{Z}_p, g$) is **APX**-hard for any nonconstant g and prime p. Note that if g is a constant function then every feasible solution has the same measure. Hence, in this case an optimal solution can be found in polynomial time.

We recall the structure theorem for abelian groups: For a finite abelian group $G = (D; +, -)$ we have

$$G \cong \mathbb{Z}_{p_1^{\alpha_1}} \times \cdots \times \mathbb{Z}_{p_n^{\alpha_n}} \tag{4.1}$$

for some integer n, primes $p_1, \ldots, p_n$ and integers $\alpha_1, \ldots, \alpha_n$. See e.g. Theorem 11.3 in [98]. In other words, any finite abelian group is isomorphic to a direct product of the form above. In the subsequent parts of this chapter we will assume that, unless explicitly stated otherwise, the group G is defined as above. We will also identify the group with its domain, i.e., we will sometimes treat G as a set such that $G = D$. We see the elements in G as vectors of integers. Position number i in each such vector is an element of $\mathbb{Z}_{p_i^{\alpha_i}}$.

4.2 Inapproximability

In this section we are going to prove inapproximability results for W-MAX SOL EQN. Johan Håstad [81] has proved an inapproximability result for a certain MAX CSP called MAX-Ek-LIN-G (we will define this problem in Section 4.2.1), which will serve as the starting point for our inapproximability result. In Section 4.2.2 we prove a first inapproximability result for W-MAX SOL EQN(G, g). This bound turns out to be tight for some groups G and some functions $g : G \to \mathbb{N}$, but not for all such combinations. We will then use this result as a stepping stone to prove our final inapproximability result in Section 4.2.3. The proof of the final result relies on the observation that for some combinations of groups, G, and functions, g, it is possible to construct a linear system of equations that induce a subgroup of G which is, in a sense made clear below, hard to approximate.

This latter result is the hardness part of Theorem 4.3 and is the main result of this section. It is formally stated as follows:

Theorem 4.4. *For every finite abelian group G and every nonconstant function $g : G \to \mathbb{N}$ it is not possible to approximate* W-MAX SOL EQN(G, g) *within $\alpha - \epsilon$ where*

$$\alpha = \max \left\{ |B| \cdot \frac{g_{\max}(B)}{g_{\text{sum}}(B)} \;\middle|\; B \in \text{EQN-COSET}(G) \right\}$$

for any $\epsilon > 0$ unless $\mathbf{P} = \mathbf{NP}$.

4.2.1 Preliminaries

We will prove our inapproximability results with a special kind of reduction, namely a gap-preserving reduction introduced by Arora in [5]. The definition is as follows.

Definition 4.5 (Gap-preserving reduction [5])**.** *Let Π and Π' be two maximisation problems and $\rho, \rho' > 1$. A* gap-preserving reduction with parameters c, ρ, c', ρ' *from Π to Π' is a polynomial time algorithm f. For each instance I of Π, f produces an instance $I' = f(I)$ of Π'. The optima of I and I', satisfy the following properties:*

- *if* $\text{OPT}(I) \geq c$ *then* $\text{OPT}(I') \geq c'$, *and*
- *if* $\text{OPT}(I) \leq c/\rho$ *then* $\text{OPT}(I') \leq c'/\rho'$.

Gap-preserving reductions are useful because if there is a polynomial time reduction, for every language in **NP**, to the maximisation problem Π such that YES instances are mapped to instances of Π of measure at least c and NO instances to instances of measure at most c/ρ, then a gap-preserving reduction from Π to Π' implies that finding ρ'-approximations to Π' is **NP**-hard. [5]

As mentioned above we will use the inapproximability of MAX-Ek-LIN-G as a starting point for our reductions. This problem is defined as follows.

Definition 4.6 (Max-Ek-Lin-G [81])**.** Max-Ek-Lin-G *is an optimisation problem with*

Instance: *A pair (V, E) where V is a set of variables and E a multiset of equations over G with exactly k variables in each equation.*

Solution: *An assignment $f : V \to G$*

Measure: *The number of satisfied equations.*

Note that for each integer k and group G, Max-Ek-Lin-G is a Max CSP(Γ) for a suitable Γ. The following theorem can be deduced from the proof of Theorem 5.9 in [81].[1]

Theorem 4.7. *For every integer $k \geq 3$, abelian group G, and problem Π in* **NP** *there is a polynomial time reduction from instances I of Π to instances $I' = (V, E)$ of* Max-Ek-Lin-G *such that*

- *if I is a* Yes *instance then at least $(1-\delta)|E|$ equations can be satisfied, and*
- *if I is a* No *instance then no assignment satisfies more than $|E|(1 + \delta)/|G|$ equations*

where $\delta > 0$ is an arbitrarily small constant. Furthermore, no equation in E contains any variables in their inverted form.

4.2.2 A First Inapproximability Result

In this section we prove a first inapproximability result for W-Max Sol Eqn(G, g). This result will be used in Section 4.2.3 to prove a stronger inapproximability result for W-Max Sol(G, g).

Lemma 4.8. *For any finite abelian group G and any nonconstant function $g : G \to \mathbb{N}$, it is not possible to approximate* W-Max Sol Eqn(G, g) *within $\alpha - \epsilon$ where*

$$\alpha = |G| \cdot \frac{g_{\max}}{g_{\text{sum}}}$$

for any $\epsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$.

Proof. Choose $k > 1$ such that $\gcd(k, p_i) = 1$ for every i, $1 \leq i \leq n$. We could, for example, choose k as $k = 1 + \prod_{i=1}^{n} p_i$. (Here n and $p_1, \ldots, p_n$ comes from the decomposition of G into a direct product, see (4.1).)

We will prove the theorem with a reduction from Max-Ek-Lin-G. Theorem 4.7 makes this a suitable approach. Given an instance, J, of an arbitrary

[1] In [81] the theorem is first proved for the case $k = 3$. There is then, on page 827, a hint on how this proof can be generalised to an arbitrary k. However, it appears that the proof which is suggested in [81] does not work. A slight modification of Theorem 5.9 in [81] does give the desired result, though [82].

problem Π in **NP**, reduce J to an instance, $I = (V, E)$, of MAX-Ek-LIN-G with the reduction in Theorem 4.7. We will use I to construct an instance $I' = (V, E', w')$ of W-MAX SOL EQN(G, g). According to Theorem 4.7 every equation e_j in E is of the form $x_1 + \ldots + x_k = c_j$. For every $e_j \in E$ add the equation e'_j, defined as $x_1 + \ldots + x_k + c_j = z_j$ to E', where z_j is a fresh variable. Let $w'(z_j) = 1$ for all $1 \leq j \leq |E|$ and $w'(\cdot) = 0$ otherwise.

We claim that the procedure presented above is a gap-preserving reduction from MAX-Ek-LIN-G to W-MAX SOL EQN(G, g) with parameters

$$\begin{aligned} c &= (1-\delta)|E|, \\ c' &= (1-\delta)|E|g_{\max}, \text{ and} \\ \rho &= |G|\frac{1-\delta}{1+\delta}, \end{aligned}$$

where δ is the constant from Theorem 4.7. The last parameter, ρ', is specified below. According to Theorem 4.7 we know that either $\text{OPT}(I) \geq |E|(1-\delta) = c$ or $\text{OPT}(I) \leq |E|(1+\delta)/|G| = c/\rho$.

Case 1: $\text{OPT}(I) \geq (1-\delta)|E|$**.** Let f be an assignment such that $m(I, f) \geq (1-\delta)|E|$. Let b be an element in G such that $g(b) = g_{\max}$ and let q be the element in G such that $kq = b$. Such a q exists because $\gcd(k, p_i) = 1$ for every i, $1 \leq i \leq n$. Construct an assignment f' as follows: let $f'(x) = f(x) + q$ for every $x \in V$ and let $f'(z_j)$ be the value in G such that equation e'_j is satisfied.

It is clear that every equation in E' is satisfied by f'. Furthermore, note that for the equations in E which are satisfied by f we have, for the corresponding equation e'_j,

$$\begin{aligned} f'(z_j) &= f'(x_1) + \ldots + f'(x_k) - c_j \\ &= kq + f(x_1) + \ldots + f(x_k) - c_j \\ &= kq = b \end{aligned}$$

Hence, for every equation e_j which is satisfied by f the variable z_j gets the value b. As $g(b) = g_{\max}$ we get

$$m'(f', I') \geq (1-\delta)|E'|g_{\max} = (1-\delta)|E|g_{\max} = c'.$$

Case 2: $\text{OPT}(I) \leq |E|(1+\delta)/|G|$**.**

For an assignment f to I let $N(f)$ denote the maximum cardinality of the set

$$\{\boldsymbol{e_j} \in E \mid \boldsymbol{e_j} \equiv x_1 + \ldots + x_k = c_j, f(x_1) + \ldots + f(x_k) - c_j = a\}$$

when a ranges over the elements of G. So $N(f)$ is the maximum number of equations which has the same "error term" under f. Let f be an assignment to I which maximises $N(f)$ and let a be the corresponding error term.

We claim that $N(f) = \text{OPT}(I)$. Note that if $a = 0_G$, then $N(f) = m(f, I) \leq \text{OPT}(I)$. To show the claim we construct a new assignment, g, to I

such that $m(I,g) = N(f)$. Let q be the element of G such that $kq = a$. Such a q exists as $\gcd(k, p_i) = 1$ for $i \in [n]$. Now, let $g(v) = f(v) - q$ for $v \in V$. If for some equation $x_1 + \ldots + x_k = c_j$ in E we have $f(x_1) + \ldots + f(x_k) - c_j = a$, then $g(x_1) + \ldots + g(x_k) - c_j = -kq + f(x_1) + \ldots + f(x_k) - c_j = 0_G$. Hence, g satisfies all equations which evaluates to a under f and the claim holds.

By the claim above it follows that for any assignment f' to I', any subset of the z_j variables that have been assigned the same value must contain at most $\lfloor |E|(1+\delta)/|G| \rfloor$ variables. (Otherwise we would have $\text{OPT}(I) > |E|(1+\delta)/|G|$, which contradicts our assumption.) The measure of any assignment, f', to I' is then bounded by

$$\begin{aligned} m'(I', f') &\leq \sum_{d \in G} \left\lfloor |E| \frac{1+\delta}{|G|} \right\rfloor g(d) \\ &\leq |E| \frac{1+\delta}{|G|} \sum_{d \in G} g(d) \\ &= |E| \frac{1+\delta}{|G|} g_{\text{sum}} \end{aligned}$$

Let h denote the quantity on the right hand side of the inequality above. We want to find the largest ρ' that satisfies $\text{OPT}(I') \leq c'/\rho'$. If we choose ρ' such that $c'/\rho' = h$, then $\text{OPT}(I') \leq c'/\rho'$ because of $\text{OPT}(I') \leq h$. Hence,

$$\begin{aligned} \rho' = \frac{c'}{h} &= \frac{(1-\delta)|E| g_{\max}}{|E| \frac{1+\delta}{|G|} g_{\text{sum}}} \\ &= |G| \cdot \frac{g_{\max}}{g_{\text{sum}}} \cdot \frac{1-\delta}{1+\delta}. \end{aligned}$$

Now, given a fixed but arbitrary $\epsilon > 0$ we can choose $0 < \delta < 1$ such that

$$\rho' > |G| \cdot \frac{g_{\max}}{g_{\text{sum}}} - \epsilon = \alpha - \epsilon.$$

Note that due to the assumption that g is nonconstant it follows that $|G| g_{\max}/g_{\text{sum}} > 1$. The gap-preserving reduction implies that it is **NP**-hard to find ρ'-approximations to W-Max Sol Eqn(G, g), and as $\rho' > \alpha - \epsilon$ we have the desired result. □

4.2.3 Inapproximability of W-Max Sol Eqn

We are now ready to prove the main inapproximability theorem.

Proof (Of Theorem 4.4). We will begin with an outline of the proof. Let I' be an arbitrary instance of W-Max Sol Eqn(G', g'), where G' is a new group and $g' : G' \to \mathbb{N}$ is a new function, both of them will soon be defined. We will then prove that W-Max Sol Eqn(G', g') is not approximable within $\alpha - \epsilon$ for any $\epsilon > 0$ unless $\mathbf{P} = \mathbf{NP}$. As the final step we will transform I' to an essentially equivalent instance I of W-Max Sol

EQN(G, g). That is, for every solution s to I we can construct a solution s' to I' in polynomial time such that $m(I, s) = m'(I', s')$ and vice versa.

If we could approximate W-MAX SOL EQN(G, g) within some performance ratio $\beta < \alpha$ we would be able to approximate W-MAX SOL EQN(G', g') within β too, because given an instance, I', of W-MAX SOL EQN(G', g') we can transform it into an essentially equivalent instance, I, of W-MAX SOL EQN(G, g) and find a β-approximate solution, s, to this instance. This solution, s, can then be transformed into a β-approximate solution, s', to I' (due to the relation between I and I', they are essentially equivalent). We will now prove the theorem.

Let A be a matrix, $\boldsymbol{b}$ be a vector, c a group constant, and $\boldsymbol{x} = (x_1, \ldots, x_m)^T$ a vector of variables such that $G' = \{x_1 \mid A\boldsymbol{x} = \boldsymbol{b}\}$ is a group and

$$B = \{x + c \mid x \in G'\} \text{ and } \alpha = |B| \cdot \frac{g_{\max}(B)}{g_{\text{sum}}(B)}.$$

The objects A, $\boldsymbol{b}$, and c do clearly exist due to the definition of equation cosets.

We define $g' : G' \to \mathbb{N}$ as $g'(x) = g(x+c)$. Note that $g'_{\max} = g_{\max}(B)$ and $g'_{\text{sum}} = g_{\text{sum}}(B)$. Hence, according to Lemma 4.8, W-MAX SOL EQN(G', g') is not approximable within $\alpha - \epsilon$ for any $\epsilon > 0$ unless $\mathbf{P} = \mathbf{NP}$.

An instance, $I' = (V', E', w')$, of W-MAX SOL EQN(G', g') can be transformed into an instance, $I = (V, E, w)$, of W-MAX SOL EQN(G, g) in the following way, assume that $V' = \{x_1, \ldots, x_{m'}\}$ and let

$$V = V' \cup \{y_{ij} \mid 1 \leq i \leq m', 1 \leq j \leq m'\} \cup \{z_i \mid 1 \leq i \leq m'\}.$$

For each variable x_i in V add the equations

$$\begin{aligned} A \begin{pmatrix} y_{i1} \\ \vdots \\ y_{im'} \end{pmatrix} &= \boldsymbol{b} \\ x_i &= y_{i1} \\ z_i &= x_i + c \end{aligned}$$

to the set E''. Those equations will force the x_i variables to be assigned values that are in G'. Finally we let $E = E' \cup E''$. The weight function, w, is constructed as follows, for $1 \leq i \leq m'$, $w(z_i) = w'(x_i)$ otherwise $w(\cdot) = 0$.

Given a solution $s : V \to G$ to I we can construct a solution $s' : V' \to G'$ to I' with the property that $m(s, I) = m'(s', I')$. Note that the equations in E'' force the x_i variables to be assigned values that are contained in G'. Hence, for every feasible solution s to I we have that $s(x_i) \in G'$ for all i such that $1 \leq i \leq m$. Let s' be constructed as $s'(x_i) = s(x_i)$ for all i such that $1 \leq i \leq m$. Note that s' must be a feasible solution to I' as we have included the equations in E' in E. The measure of s and s' are then related

to each other in the following way,

$$\begin{aligned} m(I,s) &= \sum_{i=1}^{m} w(z_i)g(s(z_i)) \\ &= \sum_{i=1}^{m} w'(x_i)g(s(x_i)+c) \\ &= \sum_{i=1}^{m} w'(x_i)g'(s'(x_i)) = m(I',s'). \end{aligned}$$

If we are given a solution, r' to I' we can in a similar way construct a solution r to I such that $m(I,r) = m'(I',r')$.

This concludes the proof, as if we can approximate W-Max Sol Eqn(G,g) within $\alpha - \epsilon$ for any $\epsilon > 0$, then we can also approximate W-Max Sol Eqn(G',g') within $\alpha - \epsilon$ in polynomial time. However, by Lemma 4.8 this is not possible unless $\mathbf{P} = \mathbf{NP}$. □

4.2.4 W-Max Sol Eqn$(\mathbb{Z}_p, g)$ is APX-complete

In this section we will prove another type of hardness result for W-Max Sol Eqn$(\mathbb{Z}_p, g)$, namely that it is **APX**-hard (for any nonconstant g and prime p). Together with the approximation algorithm in Section 4.3 the hardness result immediately implies that W-Max Sol Eqn$(\mathbb{Z}_p, g)$ is **APX**-complete. This latter result will be useful to us in Chapter 5.

We begin by giving an L-reduction from the **APX**-complete problem Max p-Cut [9] to W-Max Sol Eqn$(\mathbb{Z}_p, g)$, where p is prime. Max p-Cut is defined as follows.

Definition 4.9 (Max-p-Cut [9])**.** Max-p-Cut *is an optimisation problem with*

Instance: *A graph* $G = (V, E)$.

Solution: *A partition of* V *into* p *disjoint sets* $C_1, C_2, \ldots, C_p$.

Measure: *The number of edges between the disjoint sets, i.e.,*

$$\sum_{i=1}^{p-1} \sum_{j=i+1}^{p} |\{\{v,v'\} \in E \mid v \in C_i \text{ and } v' \in C_j\}|.$$

We will need the following simple lemma.

Lemma 4.10. *For any instance* $I = (V, E)$ *of* Max-p-Cut, $\text{opt}(I) \geq |E|(1 - 1/p)$.

Proof. Given any instance $I = (V, E)$, for each $v \in V$ choose $s(v)$ uniformly at random from $[p]$. The expected value of this solution is $|E|(1 - 1/p)$ and hence $\text{opt}(I) \geq |E|(1 - 1/p)$. □

We can now prove the **APX**-hardness of W-Max Sol Eqn$(\mathbb{Z}_p, g)$.

Lemma 4.11. *For every prime p and every nonconstant function $g : \mathbb{Z}_p \to \mathbb{N}$,* W-MAX SOL EQN$(\mathbb{Z}_p, g)$ *is* **APX***-hard.*

Proof. We give an L-reduction from MAX-p-CUT to W-MAX SOL EQN$(\mathbb{Z}_p, g)$. As MAX-p-CUT is in **APX** the lemma follows from Lemma 2.5.

Given an instance $I = (V, E)$ of MAX p-CUT, we construct an instance $F(I)$ of W-MAX SOL EQN$(\mathbb{Z}_p, g)$, where, for every vertex $v_i \in V$, we create a variable x_i and give it weight 0 and, for every edge $\{v_i, v_j\} \in E$, we create p variables $z_{ij}^{(k)}$ for $k = 0, \ldots, p-1$ and give them weight 1. Let $w \in \mathbb{Z}_p$ be a minimiser of g. For every edge $\{v_i, v_j\} \in E$, we introduce the equations

$$k(x_i - x_j) + w = z_{ij}^{(k)}$$

for $k = 0, \ldots, p-1$. If $x_i = x_j$, then the p equations for the edge $\{v_i, v_j\}$ will contribute $pg(w)$ to the measure of the solution. On the other hand, if $x_i \neq x_j$, then the p equations will contribute g_{sum} to the measure.

Given a solution s' to $F(I)$, we can construct a solution s to I in the following way: let $s(v_i) = s'(x_i)$; i.e., for every vertex v_i, place this vertex in partition $s'(x_i)$. Conversely, given a solution s to I we can construct a solution s' to $F(I)$ by $s'(x_i) = s(v_i)$ and defining $s'\left(z_{ij}^{(k)}\right)$ appropriately (note that there is a unique extension of s' to all variables in $F(I)$ which makes it a feasible solution to $F(I)$). The measures of the solutions s and s' are related to each other by the equality

$$m'(F(I), s') = m(I, s) \cdot g_{\text{sum}} + (|E| - m(I, s)) \cdot p \cdot g(w). \tag{4.2}$$

From (4.2), we get

$$\text{OPT}(F(I)) = |E| \cdot p \cdot g(w) + (g_{\text{sum}} - p \cdot g(w)) \cdot \text{OPT}(I). \tag{4.3}$$

Furthermore, from Lemma 4.10, it follows that $\text{OPT}(I) \geq |E|/p$. By combining this with (4.3), we can conclude that

$$\begin{aligned} \text{OPT}(F(I)) &= \text{OPT}(I)\left(\frac{|E| \cdot p \cdot g(w)}{\text{OPT}(I)} + g_{\text{sum}} - p \cdot g(w)\right) \\ &\leq \text{OPT}(I)\Big(p^2 \cdot g(w) + g_{\text{sum}} - p \cdot g(w)\Big). \end{aligned}$$

Hence, $\beta = p(p-1) \cdot g(w) + g_{\text{sum}}$ is an appropriate parameter for the L-reduction. We will now deduce an appropriate γ-parameter for the L-reduction: from (4.2) and (4.3) we get

$$|\text{OPT}(F(I)) - m'(F(I), s')| = (g_{\text{sum}} - p \cdot g(w)) \cdot |\text{OPT}(I) - m(I, s)|;$$

thus, $\gamma = g_{\text{sum}} - p \cdot g(w)$ is sufficient. (Note that $\gamma > 0$ because a nonconstant g implies $g_{\text{sum}} > p \cdot g(w)$). □

4.3 Approximability

In this section we will give a polynomial-time approximation algorithm for W-Max Sol Eqn(G, g). Our algorithm uses an algorithm for solving systems of equations over G in polynomial time as a subroutine. For abelian G this can be considered a folklore result and we refer the reader to [69] for a description of such an algorithm. (As mentioned in the introduction to this chapter it is **NP**-hard to find feasible solutions to the problem for non-abelian groups, see [69].)

Theorem 4.12. *For every finite abelian group G and function $g : G \to \mathbb{N}$,* W-Max Sol Eqn(G, g) *can be approximated within α in polynomial time where*

$$\alpha = \max\left\{ |B| \cdot \frac{g_{\max}(B)}{g_{\text{sum}}(B)} \;\middle|\; B \in \text{Eqn-Coset}(G) \right\}.$$

Proof. Let $I = (V, E, w)$ be an arbitrary instance of W-Max Sol Eqn(G, g) where $V = \{v_1, \ldots, v_n\}$. Feasible solutions to this optimisation problem can be viewed as certain elements in $H = G^n$. Each equation $E_i \in E$ defines a coset $a_i + J_i$ of H with representative $a_i \in H$, for some subgroup J_i of H. The set of solutions to the problem is the intersection of all those cosets. Thus, $S = \bigcap_{i=1}^{|E|} a_i + J_i$ denotes the set of all solutions. Clearly, S is empty if and only if there are no solutions. It is well-known that an intersection of a set of cosets is either empty or a coset so if $S \neq \emptyset$, then S is a coset.

We will represent the elements of G^n by vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$ where each x_i is an element of G. For any instance I, we define $\mathcal{R}(I)$ to be the random variable which is uniformly distributed over the set of solutions to I. Let V_i denote the random variable which corresponds to the value which will be assigned to v_i by $\mathcal{R}(I)$. We claim that V_i is uniformly distributed over some subset of G. As S is a coset there is a subgroup S' of G^n and an element $\boldsymbol{s} \in S$ such that $S = \boldsymbol{s} + S'$. Assume, for the sake of contradiction, that V_i is not uniformly distributed. Then, there are group elements $a, b \in G$ such that the sets

$$X_a = \{\boldsymbol{x} \in S' \mid x_i = a\} \text{ and } X_b = \{\boldsymbol{x} \in S' \mid x_i = b\}$$

have different cardinality. Assume that $|X_a| > |X_b|$. Arbitrarily pick $\boldsymbol{y} \in X_a, \boldsymbol{z} \in X_b$ and construct the set $Z = \{\boldsymbol{x} - \boldsymbol{y} + \boldsymbol{z} \mid \boldsymbol{x} \in X_a\}$. From the definition of Z and the fact that S is closed under $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \mapsto \boldsymbol{x} - \boldsymbol{y} + \boldsymbol{z}$, it follows that $Z \subseteq S$. For each $\boldsymbol{x} \in Z$ we have $x_i = b$, hence $Z \subseteq X_b$. However, we also have $|Z| = |X_a|$, which contradicts the assumption that $|X_a| > |X_b|$. We conclude that this cannot hold and V_i is uniformly distributed. Hence, for each $1 \leq i \leq n$, V_i is uniformly distributed.

For $i \in [n]$ let Y_i be the set of values which are possible for v_i. That is, $Y_i = \{x \in G \mid \Pr[V_i = x] > 0\}$. Let $S^* = \sum_{i \in [n]} w(v_i) g_{\max}(Y_i)$ and note that $S^* \geq \text{opt}(I)$. The expected value of the measure of $\mathcal{R}(I)$ can now be

estimated as

$$\begin{aligned} \mathrm{E}\left[\sum_{i=1}^{n} w(v_i)g(V_i)\right] &= \sum_{i=1}^{n} w(v_i)\mathrm{E}\left[g(V_i)\right] \\ &= \sum_{i=1}^{n} w(v_i)\frac{g_{\mathrm{sum}}(Y_i)}{|Y_i|} \\ &\geq \frac{1}{\alpha}\cdot S^* \geq \frac{1}{\alpha}\cdot \mathrm{OPT}(I). \end{aligned} \tag{4.4}$$

The first inequality follows by comparing the sums term by term

$$w(v_i)\frac{g_{\mathrm{sum}}(Y_i)}{|Y_i|} \geq \frac{1}{\alpha}\cdot w(v_i)g_{\max}(Y_i) \iff \alpha \geq |Y_i|\cdot\frac{g_{\max}(Y_i)}{g_{\mathrm{sum}}(Y_i)}.$$

The latter inequality holds by the definition of α and hence the measure of $\mathcal{R}(I)$ has, in expectation, a constant performance ratio (note that α does not depend on I).

To get a deterministic polynomial-time algorithm, note that for any instance I we can compute the sum in (4.4) in polynomial-time. Hence, we can compute the expected measure of $\mathcal{R}(I)$ in polynomial-time. Our algorithm is presented as Algorithm 4.1.

Algorithm 4.1: The algorithm in Theorem 4.12.

Input: An instance $I = (V, E, w)$ of W-MAX SOL EQN(G, g)
Output: A solution with performance ratio at least α, or "no solution" if there are no solutions.

```
1  Return "no solution" if there are no solutions.
2  Let I_1 = I
3  For i from 1 to |V| do
4      For each x ∈ G do
5          Add the constraint v_i = x to I_i.
6          If there is no solution to I_i, then go to 8.
7          Compute the expected measure of R(I_i).
8          Remove the constraint v_i = x from I_i.
9      end
10     Let x_i ∈ G be the value which maximises the expected measure of
       R(I_i) in the computations in 4–9. Create a new instance, I_{i+1},
       which is identical to I_i except for the addition of the constraint
       v_i = x_i.
11 end
12 Return the unique solution to I_{|V|+1}.
```

We claim that the following loop invariant holds in the algorithm: before line 4 is executed it is always the case that the expected measure of $\mathcal{R}(I_i)$ is at least $\mathrm{OPT}(I)/\alpha$.

We first prove the correctness of the algorithm assuming that the loop invariant holds. From the loop invariant it follows that the expected measure of $\mathcal{R}(I_{|V|+1})$ is at least $\textsc{opt}(I)/\alpha$. In $I_{|V|+1}$ there is, for each variable $v_i \in V$, a constraint of the form $v_i = x_i$, therefore there is only one solution to $I_{|V|+1}$. This solution will be returned by the algorithm.

We now prove that the loop invariant holds. The first time line 4 is reached the expected performance ratio of $\mathcal{R}(I_1)$ is at least $\textsc{opt}(I)/\alpha$, per the calculations above. Now assume that the loop invariant holds in iteration $i = k \leq |V|$; we will prove that it also holds in iteration $i = k + 1$. Since the performance ratio of $\mathcal{R}(I_k)$ is at least $\textsc{opt}(I)/\alpha$, there must be some value $x \in G$ such that when v_i is fixed to x, the performance ratio of $\mathcal{R}(I_{k+1})$ is at least $\textsc{opt}(I)/\alpha$. This element will be found by the algorithm as it maximises the expected measure of $\mathcal{R}(I_{k+1})$. Hence, the loop invariant holds for $i = k + 1$. □

Chapter 5

Max Sol on Maximal Constraint Languages

5.1 Introduction

A constraint language Γ is *maximal* if $\langle\Gamma\rangle \neq R_D$ and for any $R \notin \langle\Gamma\rangle$ we have $\langle\Gamma \cup \{R\}\rangle = R_D$. That is, the maximal constraint languages are the largest constraint languages that are not able to express all finitary relations over D. In this chapter the complexity of Max Sol(Γ) is studied for maximal constraint languages Γ.

Maximal constraint languages have attracted some attention: for instance, the complexity of the corresponding CSP has been completely classified. In [26] the complexity was classified for domains $|D| \leq 3$, and necessary conditions for tractability were proved for the general case. More recently, in [20], it was proved that those necessary conditions also are sufficient for tractability. Maximal constraint languages have also been studied in the context of machine learning [48] and quantified CSPs [31].

We begin by proving that the algebraic approach, which we gave an outline of in Section 2.2, is applicable to W-Max Sol. This result can be found in Theorem 5.5.[1] In fact, we show that, given two constraint languages Γ_1 and Γ_2 such that $\langle\Gamma_1\rangle = \langle\Gamma_2\rangle$, W-Max Sol($\Gamma_1$) S-reduces to W-Max Sol(Γ_2), and vice versa. As the S-reduction is such a strong approximation preserving reduction we get that if $\langle\Gamma_1\rangle = \langle\Gamma_2\rangle$, then Γ_1 and Γ_2 are very similar with respect to approximability. For instance, if W-Max Sol(Γ_1) is **NP**-hard to approximate within some constant c, then W-Max Sol(Γ_2) is **NP**-hard to approximate within c, too. We note that the clone theoretic approach was not used in the original classification of Max Ones

[1] The proof is easy to adapt to other problems such as W-Min Sol (the minimisation version of W-Max Sol) and Max AW Sol (where both positive and negative weights are allowed).

and, consequently, the technique we use differs substantially from those used in [102].

Our results show that if Γ is maximal and $|D| \leq 4$, then W-MAX SOL(Γ) is tractable, or **APX**-complete, or **poly-APX**-complete, or finding any solution with nonzero measure is **NP**-hard, or CSP(Γ) is not tractable. Moreover, we prove that under a conjecture by Szczepara [137] (see Conjecture 5.2) our classification of maximal constraint languages extends to arbitrary finite domains.[2] We also note that the different cases can be efficiently recognised; i.e., for a fixed domain D the approximability of a maximal constraint language $\Gamma \subseteq R_D$ can be decided in polynomial time in the size of Γ.

When proving this result, we identified a new large tractable class of W-MAX SOL(Γ): *generalised max-closed* constraints. This class (which may be of independent interest) significantly extends some of the tractable classes of MAX ONES that were identified by Khanna et al. [102]. It is also related to *monotone* constraints which have been studied in mathematical programming and computer science [77, 78, 141]. In fact, generalised max-closed constraints generalise monotone constraints over finite domains.

As in Chapter 4 we can use the results from [102] to get strong connections between the approximability of the weighted and unweighted variants of MAX SOL. Recall that from [102] it is known that the approximability of the weighted and unweighted versions of (W)-MAX SOL coincide for all boolean constraint languages. The same result holds for all constraint languages considered in this chapter. This can be readily verified by observing that the *AP*-reduction from W-MAX ONES to MAX ONES in the proof of Lemma 3.11 in [102] easily generalises to arbitrary finite domains. Hence, we get an *AP*-reduction from W-MAX SOL to MAX SOL. Furthermore, our tractability proofs are given for the weighted version of the problem. In general, it is still an open problem if tractability (i.e., membership in **PO**) of MAX SOL(Γ) implies tractability of W-MAX SOL(Γ) for every constraint language Γ (the *AP*-reduction used above does not give us this result, as *AP*-reductions do not, in general, preserve membership in **PO**).

The chapter is structured as follows: In Section 5.2 our results are presented formally, Section 5.3 presents how the algebraic approach which has been used to study CSP can be used for MAX SOL as well, Section 5.4 identifies certain hard constraint languages, and Section 5.5 contains some tractability results. In Section 5.6 we study the complexity of MAX SOL for four types of operations which will be important for our main result. Finally, in Section 5.7 we assemble the different pieces and prove our classification result for maximal constraint languages.

[2] In a conference paper [96], which this chapter is based on, we claimed that we had characterised the complexity of MAX SOL for all maximal constraint languages. Unfortunately, there was a flaw in one of the proofs. We have managed to repair some of it by proving the weaker results as stated here, but the general case, when $|D| > 4$ and Szczepara's conjecture is not assumed to hold, remains open.

5.2 Description of Results

We will now describe the results we will obtain in this chapter formally. To do so, we need a number of operations: an operation f over D is said to be

- a *constant* operation if f is unary and there is $c \in D$ such that $f(a) = c$ for all $a \in D$;
- a *majority* operation if f is ternary and $f(a,a,b) = f(a,b,a) = f(b,a,a) = a$ for all $a, b \in D$;
- a *binary commutative idempotent* operation if f is binary, $f(a,a) = a$ for all $a \in D$, and $f(a,b) = f(b,a)$ for all $a, b \in D$;
- an *affine* operation if f is ternary and $f(a,b,c) = a - b + c$ for all $a, b, c \in D$, where $+$ and $-$ are the binary operations of an abelian group $(D, +, -)$.

As mentioned in the introduction a *maximal constraint language* Γ is a constraint language such that $\langle\Gamma\rangle \subset R_D$, and if $R \notin \langle\Gamma\rangle$, then $\langle\Gamma \cup \{R\}\rangle = R_D$. This implies, among other things, that there exists an operation f such that $\langle\Gamma\rangle = \mathrm{Inv}(f)$ whenever Γ is a maximal constraint language [128]. Relational clones $\langle\Gamma\rangle$ where Γ is a maximal constraint language are called maximal relational clones. The complexity of the CSP(Γ) problem for all maximal constraint languages on domains $|D| \leq 3$ was determined in [26]. Moreover, it was shown in [26] that the only case that remained to be classified in order to extend the classification to all maximal constraint languages over a finite domain was the case where $\langle\Gamma\rangle = \mathrm{Inv}(f)$ for binary commutative idempotent operations f. These constraint languages were finally classified by Bulatov [20].

Theorem 5.1 (See [20, 26])**.** *Let Γ be a maximal constraint language on an arbitrary finite domain D. Then,* CSP(Γ) *is in* **P** *if $\langle\Gamma\rangle = \mathrm{Inv}(f)$ where f is a constant operation, a majority operation, an affine operation, or a binary commutative idempotent operation. Otherwise,* CSP(Γ) *is* **NP***-complete.*

In this chapter, we will classify the approximability of W-Max Sol(Γ) for all maximal constraint languages Γ over $|D| \leq 4$. Moreover, we prove that the only cases that remain to be classified, in order to extend the classification to all maximal constraint languages over finite domains, are constraint languages Γ such that $\langle\Gamma\rangle$ is invariant under a binary commutative idempotent operation. We also prove that if a certain conjecture regarding minimal clones generated by binary operations, due to Szczepara [137], holds, then our classification can be extended to also capture these last cases.

We first state the conjecture and then our theorem. Given a binary operation f on D, the *fixity* of f is denoted $\mathcal{F}(f)$ and is defined by

$$\mathcal{F}(f) = \{(x,y) \in D^2 \mid f(x,y) \in \{x,y\}\}.$$

Conjecture 5.2 (Conjecture 131 in [137])**.** *Let D be a finite set and let f and f' be binary commutative operations on D. If* $\mathrm{Inv}(f)$ *is a maximal relational clone and* $\mathrm{Inv}(f') = \mathrm{Inv}(f)$*, then* $|\mathcal{F}(f)| = |\mathcal{F}(f')|$*.*

Although Conjecture 5.2 is not known to hold in the general case, it has been verified for small domains. In particular, it was shown in [137] that for domains D such that $|D| \leq 4$ the conjecture holds.

Our classification theorem for the complexity of W-MAX SOL is Theorem 5.3 below. Case 1 in the theorem talks about what happens when the constraint language is closed under a *generalised* max-*closed* operation. These operations are a special kind of binary operations and are defined in Section 5.5.

Theorem 5.3. *Let Γ be maximal constraint language on a finite domain D, with $|D| \leq 4$, and $\langle\Gamma\rangle = \mathrm{Inv}(f)$.*

1. *If Γ is generalised* max-*closed or if f is a constant operation such that $f(x) = \max D$ for all $x \in D$, then* W-MAX SOL(Γ) *is in* **PO***;*
2. *else if f is an affine operation, a constant operation different from the constant 0 operation, or a binary commutative idempotent operation satisfying $f(0, b) > 0$ for all $b \in D \setminus \{0\}$ (assuming $0 \in D$), or if $0 \notin D$ and f is a binary commutative idempotent operation or a majority operation, then* W-MAX SOL(Γ) *is* **APX***-complete;*
3. *else if f is a binary commutative idempotent operation (and thus $0 \in D$ and there is some $b \in D$ such that $f(0, b) = 0$) or a majority operation, then* W-MAX SOL(Γ) *is* **poly-APX***-complete;*
4. *else if f is the constant 0 operation, then finding a solution with nonzero measure is* **NP***-hard;*
5. *otherwise, finding a feasible solution is* **NP***-hard.*

Moreover, if Conjecture 5.2 holds, then the results above hold for arbitrary finite domains D.

The proof of Theorem 5.3 consists of a careful analysis of the approximability of W-MAX SOL(Γ) for all maximal constraint languages Γ such that $\langle\Gamma\rangle = \mathrm{Inv}(f)$, where f is one of the types of operations in Theorem 5.1. The most problematic case is when $\langle\Gamma\rangle = \mathrm{Inv}(f)$ for some binary commutative idempotent operation f such that there is some two element $B \in \mathrm{Inv}(f)$. Our inability to get control over these operations is the reason for why we have not managed to prove Theorem 5.3 unconditionally and without any restrictions on D. However, for some of these operations we are able to prove some results. In particular, in Section 5.6.4.2 we will show the following theorem. Recall that a *2-semilattice operation* f is a binary operation satisfying the conditions $f(x, x) = x$, $f(x, y) = f(y, x)$, and $f(x, f(x, y)) = f(x, y)$ for all $x, y \in D$.

Theorem 5.4. *Let Γ be a maximal constraint language on a finite domain D such that $\langle\Gamma\rangle = \mathrm{Inv}(f)$ for some 2-semilattice operation f.*

- *If for all $a, b \in D$ such that $f(a,b) \in \{a,b\}$ we have $f(a,b) = \max\{a,b\}$, then* W-Max Sol(Γ) *is in* **PO**;
- *else if there exist $a, b \in D$ such that $a < b$, $f(a,b) = a$, and $a^* > 0$, where a^* is the minimal element such that there is a b^* with $a^* < b^*$ and $f(a^*, b^*) = a^*$, then* W-Max Sol(Γ) *is* **APX**-*complete;*
- *otherwise $f(0,b) = 0$ for some $b > 0$ and* W-Max Sol(Γ) *is* **poly-APX**-*complete.*

5.3 Algebraic Approach to Max Sol

The following theorem states that when we are studying the approximability of W-Max Sol(Γ), it is sufficient to consider constraint languages that are relational clones.

Theorem 5.5. *Let Γ' be a constraint language and let $\Gamma \subseteq \langle\Gamma'\rangle$ be finite. Then* W-Max Sol(Γ) *is S-reducible to* W-Max Sol(Γ').

Proof. Consider an instance $I = (V, D, C, w)$ of W-Max Sol(Γ). We transform I into an instance $F(I) = (V', D, C', w')$ of W-Max Sol(Γ').

For every constraint $c = ((v_1, \ldots, v_m), R)$ in I, R can be represented as

$$\exists_{v_{m+1}}, \ldots, \exists_{v_n} R_1(v_{11}, \ldots, v_{1n_1}) \wedge \cdots \wedge R_k(v_{k1}, \ldots, v_{kn_k}),$$

where $R_1, \ldots, R_k \in \Gamma' \cup \{EQ_D\}$, $v_{m+1}, \ldots, v_n$ are fresh variables, and $v_{11}, \ldots, v_{1n_1}$, $v_{21}, \ldots, v_{kn_k} \in \{v_1, \ldots, v_n\}$. Replace the constraint c with the constraints

$$((v_{11}, \ldots, v_{1n_1}), R_1), \ldots, ((v_{k1}, \ldots, v_{kn_k}), R_k),$$

add $v_{m+1}, \ldots, v_n$ to V, and extend w so that $v_{m+1}, \ldots, v_n$ are given weight 0. If we repeat the same reduction for every constraint in C, then it results in an equivalent instance of W-Max Sol($\Gamma' \cup \{EQ_D\}$).

For each equality constraint $((v_i, v_j), EQ_D)$, we do the following:

- replace all occurrences of v_j with v_i, update w' so that the weight of v_j is added to the weight of v_i, remove v_j from V, and remove the weight corresponding to v_j from w'; and
- remove $((v_i, v_j), EQ_D)$ from C.

The resulting instance $F(I) = (V', D, C', w')$ of W-Max Sol(Γ') has the same optimum as I (i.e., $\mathrm{opt}(I) = \mathrm{opt}(F(I))$) and has been obtained in polynomial time.

Now, given a feasible solution s' for $F(I)$, let $G(I, s')$ be the feasible solution for I where the following hold:

- the variables in I assigned by s' inherit their value from s'; and
- the variables in I which are still unassigned all occur in equality constraints, and their values can be found by simply propagating the values of the variables which have already been assigned.

It should be clear that $m(I, G(I, s')) = m(F(I), s')$ for any feasible solution s' for $F(I)$. Hence, the functions F and G, as described above, are the two parts of an S-reduction from W-MAX SOL(Γ) to W-MAX SOL(Γ'). □

5.4 Hardness and Membership Results

In this section, we first prove some general **APX** and **poly-APX** membership results for W-MAX SOL(Γ). We also prove **APX**-completeness and **poly-APX**-completeness for some particular constraint languages. Most of our hardness results in the subsequent sections are based on these results.

We begin by making the following easy but interesting observation: we know from the classification of W-MAX SOL(Γ) over the boolean domain that there exist many constraint languages Γ for which W-MAX SOL(Γ) is **poly-APX**-complete. However, if 0 is not in the domain, then there are no constraint languages Γ such that W-MAX SOL(Γ) is **poly-APX**-complete.

Lemma 5.6. *If* CSP(Γ) *is in* **P** *and* $0 \notin D$*, then* W-MAX SOL(Γ) *is in* **APX**.

Proof. It is proved in [32] that if CSP(Γ) is in **P**, then we can also find a solution in polynomial time. It should be clear that this solution is a $\frac{\max(D)}{\min(D)}$-approximate solution. Hence, we have a trivial approximation algorithm with performance ratio $\frac{\max(D)}{\min(D)}$. □

Next, we present a general membership result for W-MAX SOL(Γ). The proof is similar to the proof of the corresponding result for the boolean domain [102, Lemma 6.2] so we omit the proof. For any $a \in D$, we denote the unary constant relation containing only a by c_a, i.e., $c_a = \{(a)\}$. Let C_D denote the set of all constant relations over D, that is, $C_D = \{c_a \mid a \in D\}$.

Lemma 5.7. *Let* Γ *be a constraint language over the domain* D*. If* CSP($\Gamma \cup C_D$) *is in* **P***, then* W-MAX SOL($\Gamma \cup C_D$) *is in* **poly-APX**.

We continue by proving the **APX**-completeness of some constraint languages.

Lemma 5.8. *Let* $R = \{(a, a), (a, b), (b, a)\}$ *and* $a, b \in D$ *such that* $0 < a < b$*. Then,* W-MAX SOL($\{R\}$) *is* **APX**-*complete.*

Proof. Containment in **APX** follows from Lemma 5.6. To prove the hardness result we give an L-reduction (with parameters $\beta = 4b$ and $\gamma = \frac{1}{b-a}$) from the **APX**-complete problem INDEPENDENT SET restricted to degree

3 graphs [1] to MAX SOL($\{R\}$). Given an instance $I = (V, E)$ of INDEPENDENT SET (restricted to graphs of degree at most 3 and containing no isolated vertices), let $F(I) = (V, D, C)$ be the instance of MAX SOL($\{R\}$) where, for each edge $(v_i, v_j) \in E$, we add the constraint $R(x_i, x_j)$ to C. For any feasible solution s' for $F(I)$, let $G(I, s')$ be the solution for I where all vertices corresponding to variables assigned b in s' form the independent set. We have $|V|/4 \leq \text{OPT}(I)$ and $\text{OPT}(F(I)) \leq b|V|$, so $\text{OPT}(F(I)) \leq 4b\text{OPT}(I)$. Thus, $\beta = 4b$ is an appropriate parameter.

Let K be the number of variables being set to b in an arbitrary solution s' for $F(I)$. Then,

$$|\text{OPT}(I) - m(I, G(I, s'))| = \text{OPT}(I) - K \quad \text{and}$$
$$|\text{OPT}(F(I)) - m(F(I), s')| = (b - a)(\text{OPT}(I) - K).$$

Hence,

$$|\text{OPT}(I) - m(I, G(I, s'))| = \frac{1}{b-a}|\text{OPT}(F(I)) - m(F(I), s')|,$$

and $\gamma = \frac{1}{b-a}$ is an appropriate parameter. □

The following **poly-APX**-completeness result will be useful to us several times later on.

Lemma 5.9. *If $R = \{(0,0),(0,b),(b,0)\}$ where $b \in D$ and $0 < b$, then* W-MAX SOL($\{R\}$) *is* **poly-APX***-complete.*

Proof. It is proved in [102, Lemma 6.15] that for $Q = \{(0,0),(0,1),(1,0)\}$, W-MAX SOL($\{Q\}$) is **poly-APX**-complete. To prove the **poly-APX**-hardness we give an *AP*-reduction from W-MAX SOL($\{Q\}$) to W-MAX SOL($\{R\}$). Given an instance I of W-MAX SOL($\{Q\}$), let $F(I)$ be the instance of W-MAX SOL($\{R\}$) where all occurrences of Q have been replaced by R. For any feasible solution s' for $F(I)$, let $G(I, s')$ be the solution for I where all variables assigned b in s' are instead assigned 1. It should be clear that this is an *AP*-reduction, since if s' is an α-approximate solution to $F(I)$, then $G(I, s')$ is an α-approximate solution for I.

To see that W-MAX SOL($\{R\}$) is in **poly-APX**, let $D = \{d_1, \ldots, d_n\}$ and note that $\Gamma \cup C_D = \{R, \{(d_1)\}, \ldots, \{(d_n)\}\}$ is invariant under the min function. As the min function is associative, commutative, and idempotent, CSP($\Gamma \cup C_D$) is solvable in polynomial time [90]. Hence, W-MAX SOL($\{R\}$) is in **poly-APX** due to Lemma 5.7. □

5.5 Generalised Max-closed Relations

In this section, we present tractability results for *generalised max-closed* constraint languages. This class can be seen as substantial and nontrivial generalisations of two of the tractable classes known for the corresponding W-MAX ONES problem over the boolean domain. There are only three

tractable classes of constraint languages over the boolean domain, namely, width-2 affine, 1-valid, and weakly positive [102]. Width-2 affine relations can be expressed as systems of linear equations over $\mathbb{Z}_2$ with at most two variables in each equation. 1-valid relations contain the tuple $(1, 1, \ldots, 1)$. Finally, a relation is weakly positive if it can be expressed as a formula in conjunctive normal form having at most one negated variable in each clause. Such relations are also invariant under max.

The classes of 1-valid and weakly positive constraint languages are examples of generalised max-closed constraint languages. The monotone constraints which are, for instance, studied by Hochbaum et al. [77] and Hochbaum and Naor [78] (in relation to integer programming) and Woeginger [141] (in relation to constraint satisfaction) are also related to generalised max-closed constraints. Hochbaum and Naor [78] show that monotone constraints can be characterised as those constraints that are simultaneously invariant under the max and min operators. Hence, monotone constraints are also generalised max-closed constraints as long as the underlying domain is finite.

We begin by giving the following basic definition.

Definition 5.10 (Generalized max-closed)**.** *A constraint language* Γ *over a domain* $D \subset \mathbb{N}$ *is* generalised max-closed *if and only if there exists a binary operation* $f \in \mathrm{Pol}(\Gamma)$ *such that for all* $a, b \in D$*,*

1. *if* $a \neq b$ *and* $f(a, b) \leq \min(a, b)$*, then* $f(b, a) > \max(a, b)$*; and*
2. $f(a, a) \geq a$*.*

In the conference paper which this chapter is based on [96] the definition of generalised max-closed constraint languages was slightly more restrictive. The following two examples will clarify the definition above.

Example 5.11 (Generalized max-closed relations)**.** *Assume that the domain* D *is* $\{0, 1, 2, 3\}$*. As an example of a generalised* max*-closed relation consider*

$$R = \{(0, 0), (1, 0), (0, 2), (1, 2)\}.$$

R *is invariant under* max *and is therefore generalised max-closed since* max *satisfies the properties of Definition 5.10. Now, consider the relation* Q *defined as*

$$Q = \{(0, 1), (1, 0), (2, 1), (2, 2), (2, 3)\}.$$

Q *is not invariant under* max *because* $(0, 1), (1, 0) \in Q$ *and*

$$\max((0, 1), (1, 0)) = (\max(0, 1), \max(1, 0)) = (1, 1) \notin Q.$$

Let the operation $\circ : D^2 \to D$ *be defined by the following Cayley table*

$\circ$	0	1	2	3
0	0	2	2	3
1	2	1	2	2
2	2	2	2	3
3	3	2	3	3

.

Now, it is easy to verify that $\mathrm{Inv}(\circ)$ *is a set of generalised max-closed relations and that* $Q \in \mathrm{Inv}(\circ)$.

Example 5.12 (More generalized max-closed relations)**.** *Consider the relations* R_1 *and* R_2 *defined as*

$$R_1 = \{(1,1,1),(1,0,0),(0,0,1),(1,0,1)\}$$

and $R_2 = R_1 \setminus \{(1,1,1)\}$ *over the domain* $D = \{0,1\}$*. The relation* R_1 *is* 1*-valid because the all-*1 *tuple is in* R_1*, i.e.,* $(1,1,1) \in R_1$*.* R_2*, on the other hand, is not* 1*-valid but is weakly positive because it is invariant under* max*. Note that both* R_1 *and* R_2 *are generalised max-closed since* R_1 *is invariant under* $f(x,y) = 1$ *and* R_2 *is invariant under* $f(x,y) = \max(x,y)$*. Hence, the* 1*-valid and weakly positive relations are subsets of the generalised max-closed relations.*

The tractability of generalised max-closed constraint languages depends on the following lemma.

Lemma 5.13. *If* Γ *is generalised* max*-closed, then all* m*-ary relations* R *in* $\langle\Gamma\rangle$ *have the property that the tuple*

$$(\max \mathrm{pr}_1\ R, \max \mathrm{pr}_2\ R, \ldots, \max \mathrm{pr}_m\ R)$$

is in R*, too.*

Proof. Assume, for the sake of contradiction, that there is an m-ary relation R in $\langle\Gamma\rangle$ such that the tuple

$$\boldsymbol{t}_{\mathbf{max}} = (\max \mathrm{pr}_1\ R, \max \mathrm{pr}_2\ R, \ldots, \max \mathrm{pr}_m\ R)$$

is not contained in R. Define the distance between two tuples to be the number of coordinates where they disagree (i.e., the Hamming distance). Let $\boldsymbol{a}$ be a tuple in R with minimal distance from $\boldsymbol{t}_{\mathbf{max}}$ and let I denote the set of coordinates where $\boldsymbol{a}$ agree with $\boldsymbol{t}_{\mathbf{max}}$. By the assumption that $\boldsymbol{t}_{\mathbf{max}}$ is not in R, we know that the distance between $\boldsymbol{a}$ and $\boldsymbol{t}_{\mathbf{max}}$ is at least 1. Hence, without loss of generality, assume that $\boldsymbol{a}(1) \neq \boldsymbol{t}_{\mathbf{max}}(1)$ and that $\boldsymbol{a}(1)$ is maximal for all tuples in R agreeing with $\boldsymbol{t}_{\mathbf{max}}$ on the coordinates in I. Let $\boldsymbol{b}$ be a tuple in R such that $\boldsymbol{b}(1) = \boldsymbol{t}_{\mathbf{max}}(1)$.

Since Γ is generalised max-closed, there exists an operation $f \in \mathrm{Pol}(\Gamma)$ such that for all $a, b \in D$ $(a \neq b)$, it holds that $f(a,b) > \max(a,b)$ whenever $f(b,a) \leq \min(a,b)$. Furthermore, for all $a \in D$ it holds that $f(a,a) \geq a$. Now consider the tuple $\boldsymbol{x}^{\boldsymbol{n}}$ $(n = |D|)$ defined as follows: $\boldsymbol{x}^{\mathbf{1}} = f(\boldsymbol{a}, \boldsymbol{b})$ and

$$\boldsymbol{x}^{\boldsymbol{i+1}} = \begin{cases} f(\boldsymbol{x}^{\boldsymbol{i}}, \boldsymbol{a}) & \text{if } f(\boldsymbol{x}^{\boldsymbol{i}}(1), \boldsymbol{a}(1)) > \boldsymbol{a}(1), \\ f(\boldsymbol{a}, \boldsymbol{x}^{\boldsymbol{i}}) & \text{otherwise.} \end{cases}$$

We begin by proving that $\boldsymbol{x}^{\boldsymbol{n}}$ agrees with $\boldsymbol{a}$ on all coordinates in I. Let $\boldsymbol{z}$ be an arbitrary tuple in R. Note that for each $i \in I$ such that $\boldsymbol{z}(i) \neq \boldsymbol{a}(i)$,

it is the case that $f(\boldsymbol{a}(i), \boldsymbol{z}(i)) \leq \min(\boldsymbol{a}(i), \boldsymbol{z}(i))$ implies that $f(\boldsymbol{z}(i), \boldsymbol{a}(i)) > \max(\boldsymbol{a}(i), \boldsymbol{z}(i))$. Hence, as $\boldsymbol{a}(i) = \boldsymbol{t_{max}}(i)$, we cannot have $f(\boldsymbol{a}(i), \boldsymbol{z}(i)) \leq \min(\boldsymbol{a}(i), \boldsymbol{z}(i))$. So, for each $\boldsymbol{z} \in R$ and $i \in I$, we must have $f(\boldsymbol{a}(i), \boldsymbol{z}(i)) > \min(\boldsymbol{a}(i), \boldsymbol{z}(i))$ whenever $\boldsymbol{a}(i) \neq \boldsymbol{z}(i)$. By an analogous argument, it follows that for each $\boldsymbol{z} \in R$ and $i \in I$ we must have $f(\boldsymbol{z}(i), \boldsymbol{a}(i)) > \min(\boldsymbol{a}(i), \boldsymbol{z}(i))$ whenever $\boldsymbol{a}(i) \neq \boldsymbol{z}(i)$.

This together with the fact that $f(d, d) \geq d$ for all $d \in D$ and that $\boldsymbol{a}$ agrees with $\boldsymbol{t_{max}}$ on I implies that $\boldsymbol{x^n}$ agrees with $\boldsymbol{a}$ on I.

We now show that $\boldsymbol{x^n}(1) > \boldsymbol{a}(1)$. This follows from essentially the same argument as above. We first show that $f(\boldsymbol{a}(1), \boldsymbol{b}(1)) = \boldsymbol{x^1}(1) > \boldsymbol{a}(1)$. If $f(\boldsymbol{a}(1), \boldsymbol{b}(1)) \leq \min(\boldsymbol{a}(1), \boldsymbol{b}(1))$, then $f(\boldsymbol{b}(1), \boldsymbol{a}(1)) > \boldsymbol{b}(1)$ which is not possible since $\boldsymbol{b}(1) = \boldsymbol{t_{max}}(1)$. Hence, we must have $f(\boldsymbol{a}(1), \boldsymbol{b}(1)) = \boldsymbol{x^1}(1) > \min(\boldsymbol{a}(1), \boldsymbol{b}(1))$. Now, by the definition of $\boldsymbol{x^{i+1}}$, it follows that if $\boldsymbol{x^i}(1) > \boldsymbol{a}(1)$, then $\boldsymbol{x^{i+1}}(1) > \min(\boldsymbol{x^i}(1), \boldsymbol{a}(1)) = \boldsymbol{a}(1)$ (just note that at least one of $f(\boldsymbol{x^i}(1), \boldsymbol{a}(1))$ and $f(\boldsymbol{a}(1), \boldsymbol{x^i}(1))$ is strictly larger than $\min(\boldsymbol{x^i}(1), \boldsymbol{a}(1)) = \boldsymbol{a}(1)$). Hence, it follows by induction that $\boldsymbol{x^n}(1) > \boldsymbol{a}(1)$.

Thus, we have a contradiction with the fact that $\boldsymbol{a}(1)$ is maximal for all tuples in R agreeing with $\boldsymbol{t_{max}}$ on the coordinates in I. Hence, our assumption was wrong and $\boldsymbol{t_{max}}$ is contained in R. □

The algorithm for solving W-MAX SOL(Γ) when Γ is generalised max-closed is a simple consistency-based algorithm. The algorithm, which is based on pair-wise consistency, closely follows the algorithm for CSPs over max-closed constraint languages from [91].

We first need to introduce some terminology.

Definition 5.14 (Pairwise consistent [88])**.** *An instance of a constraint satisfaction problem $I = (V, D, C)$ is* pairwise consistent *if and only if for any pair of constraints $C_i = (s_i, R_i)$, $C_j = (s_j, R_j)$ in C, it holds that the relation resulting from projecting R_i onto the variables in $s_i \cap s_j$ equals the relation resulting from projecting R_j onto the variables in $s_i \cap s_j$; that is, $\mathrm{pr}_{s_i \cap s_j} R_i = \mathrm{pr}_{s_i \cap s_j} R_j$.*

We are now ready to prove the tractability of generalised max-closed constraint languages.

Theorem 5.15. *Let $\Gamma \subseteq R_D$ be a constraint language. If all m-ary relations R in $\langle\Gamma\rangle$ have the property that*

$$(\max \mathrm{pr}_1 R, \max \mathrm{pr}_2 R, \ldots, \max \mathrm{pr}_m R) \in R, \tag{5.1}$$

then W-MAX SOL(Γ) *is in* **PO**.

Proof. It was proved in [88] that any set of constraints can be reduced to an equivalent set of pairwise consistent constraints in polynomial time. Furthermore, it is known that the resulting set of pairwise consistent constraints are still in $\langle\Gamma\rangle$. [90].

Hence, given an instance $I = (V, D, C, w)$ of W-MAX SOL(Γ), we can assume that the constraints in C are pairwise consistent. We prove that for

pairwise consistent C, either C has a constraint with a constraint relation that does not contain any tuples (i.e., no assignment satisfies the constraint and there is no solution) or we can find the optimal solution in polynomial time.

Assume that C has no empty constraints. For each variable $v \in V$, let $s(v)$ be the maximum value allowed for that variable by some constraint C_j (where v is in the constraint scope of C_j). We will prove that $s : V \to D$ is an optimal solution to I. Obviously, if s is a solution to I, then it is the optimal solution. Hence, it is sufficient to prove that s is a solution to I.

Assume, with the aim of reaching a contradiction, that s is not a solution to I. Then, there exists a constraint C_j in C not satisfied by s. Since the constraint relation corresponding to C_j satisfies (5.1), there exists a variable v in the constraint scope of C_j such that C_j has no solution where v is assigned to $s(v)$. Note that it is essential that C_j satisfies (5.1) to rule out the possibility that there exist two variables v and v' in the constraint scope of C_j such that C_j has two solutions $t, u : V \to D$ where $t(v) = d$ and $u(v') = d'$, but C_j has no solution w where $w(v) = d$ and $w(v') = d'$. We know that there exists a constraint C_i in C having v in its constraint scope and $s(v)$ an allowed value for v. This contradicts the fact that C is pairwise consistent. Thus, s is a solution to I. □

Corollary 5.16. *If* Γ *is generalised* max*-closed, then* W-Max Sol(Γ) *is in* **PO**.

Proof. Follows from Lemma 5.13 and Theorem 5.15. □

There is an algebraic description of the constraint languages which are tractable by Theorem 5.15. We will not need this characterisation for our results, but it may be interesting anyway. We need the notion of set functions introduced in [49]. A *set function* is a function $f : 2^D \setminus \emptyset \to D$. With each set function we associate a sequence of functions $f_1, f_2, \ldots$ such that f_i is of arity i and is given by

$$f_i(x_1, x_2, \ldots, x_i) = f(\{x_1, x_2, \ldots, x_i\}).$$

We say that a constraint language is closed under a set function f if it is closed under the sequence of functions associated with f.

Definition 5.17 (Max set function)**.** *A* max set function *is a set function* $f : 2^D \setminus \emptyset \to D$ *such that* $f(X) \geq \max X$ *for all* $X \subseteq D, X \neq \emptyset$.

The following theorem gives the algebraic characterisation of the constraint languages made tractable by Theorem 5.15.

Theorem 5.18. *Let* $\Gamma \subseteq R_D$ *be a constraint language. The following are equivalent*

1. *for any* n*-ary relation* R *in* $\langle\Gamma\rangle$ *we have*

$$(\max \mathrm{pr}_1\ R, \max \mathrm{pr}_2\ R, \ldots, \max \mathrm{pr}_n\ R) \in R;$$

2. Γ *is closed under a max set function.*

Proof. We first show that 1 implies 2. We will define a set function $f : 2^D \setminus \emptyset \to D$. For a nonempty subset $S \subseteq D$ let $U(S)$ be the inclusion-wise minimal set which satisfies $S \subseteq U(S)$ and $U(S) \in \langle\Gamma\rangle$. Note that there is such an inclusion-wise minimal set as if $S \subseteq X, Y$ and $X, Y \in \langle\Gamma\rangle$, then $S \subseteq X \cap Y \in \langle\Gamma\rangle$. We let $f(S) = \max U(S)$. It is clear that f is a max set function.

Let m be a positive integer and let $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m} \in R$. We will show that

$$f(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m}) \in R. \tag{5.2}$$

For $i \in [n]$ let $S_i = \{\boldsymbol{x_1}(i), \ldots, \boldsymbol{x_m}(i)\}$ and let $U_i = U(S_i)$. As $U_i \in \langle\Gamma\rangle$ for $i \in [n]$ it follows that

$$R' = R \cap \prod_{i=1}^{n} U_i$$

is contained in $\langle\Gamma\rangle$. Note that $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m} \in R'$ and $R' \subseteq R$, hence (5.2) holds if $f(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m}) \in R'$.

By our choice of U_i and the fact that $\mathrm{pr}_i\, R' \in \langle\Gamma\rangle$ for $i \in [n]$, it follows that $\mathrm{pr}_i\, R' = U_i$ for $i \in [n]$. By the definition of f it follows that

$$f(\boldsymbol{x_1}, \ldots, \boldsymbol{x_m}) = (\max U_1, \max U_2, \ldots, \max U_n)$$

which is contained in R' by the assumption in the lemma.

To show that 2 implies 1, we simply apply f to all tuples in R. □

5.6 Classification Results for Four Types of Operations

Note that to prove the main results, Theorem 5.3 and Theorem 5.4, we only need to consider operations f on D such that $\mathrm{Inv}(f)$ is a maximal constraint language and $\mathrm{CSP}(\mathrm{Inv}(f))$ is tractable. By Theorem 5.1 this limits the possibilities of f to a constant operation, a majority operation, an affine operation, or a binary commutative idempotent operation. We will deal with each of these types of operations in the subsequent four subsections.

In Section 5.6.4.2 we will give a proof of Theorem 5.4. The proof of Theorem 5.3 is postponed till Section 5.7.

5.6.1 Constant Operation

We begin by considering maximal constraint languages that are invariant under constant operations.

Lemma 5.19. *Let $d^* = \max D$ and let C_d be a constraint language such that $\langle C_d \rangle = \mathrm{Inv}(f_d)$, where $f_d : D \to D$ satisfies $f_d(x) = d$ for all $x \in D$. Then,* W-Max Sol(C_{d^*}) *is in* **PO**, W-Max Sol(C_d) *is* **APX**-*complete if $d \in D \backslash \{d^*, 0\}$, and it is* **NP**-*hard to find a solution with nonzero measure for* W-Max Sol(C_0).

Proof. The tractability of W-Max Sol(C_{d^*}) is trivial, since the optimum solution is obtained by assigning d^* to all variables.

For the **APX**-hardness of W-Max Sol(C_d) ($d \in D \backslash \{d^*, 0\}$), it is sufficient to note that $\{(d, d), (d, d^*), (d^*, d)\}$ is in $\langle C_d \rangle$, and since $0 < d < d^*$ it follows from Lemma 5.8 that W-Max Sol(C_d) is **APX**-hard. It is easy to realise that W-Max Sol(C_d) is in **APX**, since we can obtain a $\frac{d^*}{d}$-approximate solution by assigning the value d to all variables.

The fact that it is **NP**-hard to find a solution with nonzero measure for W-Max Sol(C_0) over the boolean domain $\{0, 1\}$ is proved in [102, Lemma 6.23]. From this result it follows that it is **NP**-hard to find solutions with nonzero measure to W-Max Sol(C_0) over any domain. (Any nonzero element in D can play the role of 1 in the boolean domain.) □

5.6.2 Majority Operation

Maximal constraint languages based on majority operations are fairly easy to analyse due to the results in Section 5.4.

Lemma 5.20. *Let m be an arbitrary majority operation on D. Then,* W-Max Sol(Inv(m)) *is* **APX**-*complete if $0 \notin D$ and* **poly-APX**-*complete if $0 \in D$.*

Proof. Arbitrarily choose elements $a, b \in D$ such that $a < b$. Then, it is easy to see that $\{(a, a), (a, b), (b, a)\}$ is in Inv(m). Thus, by Lemma 5.6 and Lemmas 5.8 and 5.9, it follows that W-Max Sol(Inv(m)) is **APX**-complete or **poly-APX**-complete depending on whether 0 is in D or not. □

5.6.3 Affine Operation

We will denote the affine operation on the group G by a_G; i.e., if $G = (D, +_G, -_G)$, then $a_G(x, y, z) = x -_G y +_G z$. In this section we will prove that for every affine operation $a_G : D^3 \to D$, W-Max Sol(Inv(a_G)) is **APX**-complete. This is done in two subsections, in the first one we will show that the problem is in **APX** and in the second one we show that the problem is **APX**-hard.

We will need the following lemma, which is well-known, in the two subsections below.

Lemma 5.21. *An n-ary relation R is a coset of some subgroup of G^n if and only if a_G is a polymorphism of R.*

For completeness we give a proof of this lemma.

Proof. First assume that R is of the form $\boldsymbol{x} + H$ where H is a subgroup of G^n and $\boldsymbol{x} \in G^n$. If $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in R$, then there are $\boldsymbol{a'}, \boldsymbol{b'}, \boldsymbol{c'} \in H$ such that $\boldsymbol{a} = \boldsymbol{x} + \boldsymbol{a'}$, $\boldsymbol{b} = \boldsymbol{x} + \boldsymbol{b'}$, and $\boldsymbol{c} = \boldsymbol{x} + \boldsymbol{c'}$. Hence, as H is a subgroup of G^n we have $\boldsymbol{a'} - \boldsymbol{b'} + \boldsymbol{c'} \in H$ and thus

$$\boldsymbol{a} - \boldsymbol{b} + \boldsymbol{c} = \boldsymbol{x} + \boldsymbol{a'} - \boldsymbol{x} - \boldsymbol{b'} + \boldsymbol{x} + \boldsymbol{c'} = \boldsymbol{x} + \boldsymbol{a'} - \boldsymbol{b'} + \boldsymbol{c'} \in R.$$

So a_G is a polymorphism of R.

Now assume that a_G is a polymorphism of R. Let $\boldsymbol{x}$ be some arbitrary element of R. We claim that $H = \{\boldsymbol{y} - \boldsymbol{x} \mid \boldsymbol{y} \in R\}$ is a subgroup of G^n. To see this, let $\boldsymbol{a}, \boldsymbol{b} \in H$. Note that as R is invariant under a_G we have

$$a_G(\boldsymbol{a} + \boldsymbol{x}, \boldsymbol{b} + \boldsymbol{x}, \boldsymbol{x}) = \boldsymbol{a} - \boldsymbol{b} + \boldsymbol{x} \in R$$

which implies that $\boldsymbol{a} - \boldsymbol{b} \in H$. Similarly, $a_G(\boldsymbol{a} + \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{b} + \boldsymbol{x}) = \boldsymbol{a} + \boldsymbol{b} + \boldsymbol{x} \in R$ and hence $\boldsymbol{a} + \boldsymbol{b} \in H$ as well. This implies that H is a subgroup and that R is a coset of H with representative $\boldsymbol{x}$. □

5.6.3.1 Membership in APX

We will now prove that relations that are invariant under an affine operation give rise to problems which are in **APX**. To do this we will use a slight modification of Algorithm 4.1. Recall that Algorithm 4.1 is an approximation algorithm for W-MAX SOL EQN(G, g) with constant performance ratio. In the proof of correctness of Algorithm 4.1 essentially the only information used about the constraints (the constraints were equations in Chapter 4) was that the set of solutions to one equation is a coset of some subgroup of G^n. By Lemma 5.21 this can be used in our setting.

Theorem 5.22. *Let $G = (D; +_G, -_G)$ be a finite abelian group and let Γ be a constraint language such that $a_G \in \mathrm{Pol}(\Gamma)$. Then,* W-MAX SOL$(\Gamma)$ *is in* **APX**.

Proof. By Lemma 5.21 every relation R in Γ is a coset of some subgroup of G^{n_R}, where n_R is the arity of R.

We will now use Algorithm 4.1 to prove the theorem. To do so we need two things, the first is a polynomial-time algorithm for CSP(Γ). This was given in [90] for the case when Γ is invariant under an affine operation. Secondly, we need to specify the function g. In our case this is simply the identity function. Note that in the proof of correctness of Algorithm 4.1 (Theorem 4.12) these two properties where the only ones we used about the constraint language. It now follows from Theorem 4.12 that there is a polynomial-time approximation algorithm for MAX SOL(Γ) with performance ratio

$$\max \left\{ |C| \cdot \frac{\max C}{\sum_{c \in C} c} \;\middle|\; C \text{ is a coset of } G \right\}.$$

(We cannot restrict ourselves to the equation cosets of G here as Γ may contain cosets which are not equation cosets as well. Compare with Theorem 4.12.) We conclude that W-MAX SOL(Γ) is contained in **APX**. □

5.6.3.2 APX-hardness

The hardness proof is based on a reduction from MAX SOL EQN($\mathbb{Z}_p, g$) which we proved to be **APX**-hard in Lemma 4.11.

Theorem 5.23. W-MAX SOL(Inv(a_G)) *is* **APX***-hard for every affine operation* a_G*.*

Proof. We show that there exists a prime p and a nonconstant function $h : \mathbb{Z}_p \to \mathbb{N}$ such that W-MAX SOL EQN($\mathbb{Z}_p, h$) can be S-reduced to W-MAX SOL(Inv(a_G)). The result will then follow from Lemma 4.11.

Let p be a prime such that $\mathbb{Z}_p$ is isomorphic to a subgroup H of G. We know that such a p always exists by the fundamental theorem of finitely generated abelian groups. Let α be the isomorphism which maps elements of $\mathbb{Z}_p$ to elements of H and let $h = \alpha$. (Note that $H \subset \mathbb{N}$ since the domain is a subset of $\mathbb{N}$. Consequently, h may be viewed as a function from $\mathbb{Z}_p$ to $\mathbb{N}$.)

Let $I = (V, E, w)$ be an instance of W-MAX SOL EQN($\mathbb{Z}_p, h$) with variables $V = \{v_1, \ldots, v_n\}$ and equations $E = \{e_1, \ldots, e_m\}$. We will construct an instance $I' = (V, D, C, w)$ of W-MAX SOL(Inv(a_G)).

Let U be the unary relation for which $x \in U \iff x \in H$; this relation is in Inv(a_G) by Lemma 5.21. For every equation $E_i \in E$, there is a corresponding pair $(\boldsymbol{s_i}, R_i)$, where $\boldsymbol{s_i}$ is a list of variables and R_i is a relation in Inv(a_G) such that the set of solutions to E_i contains exactly the tuples which satisfy $(\boldsymbol{s_i}, R_i)$ (this follows from Lemma 5.21 and the fact that the set of solutions to such an equation is a coset). We can now construct C:

$$C = \{(v_i, U) \mid 1 \leq i \leq n\} \cup \{(\boldsymbol{s_i}, R_i) \mid 1 \leq i \leq m\}.$$

It is easy to see that I and I' are essentially the same in the sense that every feasible solution to I is also a feasible solution to I', and that they have the same measure. The converse is also true: every feasible solution to I' is also a feasible solution to I. Hence, we have given an S-reduction from W-MAX SOL EQN($\mathbb{Z}_p, h$) to W-MAX SOL(Inv(a_G)). As h is not constant (it is in fact injective), it follows from Lemma 4.11 that W-MAX SOL EQN($\mathbb{Z}_p, h$) is **APX**-hard. This S-reduction implies that W-MAX SOL(Inv(a_G)) is **APX**-hard. □

5.6.4 Binary Commutative Idempotent Operation

We now investigate the complexity of W-MAX SOL(Γ) for maximal constraint languages Γ satisfying $\langle\Gamma\rangle = \text{Inv}(f)$ where f is a binary commutative idempotent operation.

Let $(F; +_F, -_F, \cdot_F, 1_F)$ be a finite field of prime order $p > 2$, where $+_F, -_F, \cdot_F$, and 1_F denote addition, subtraction, multiplication, and multiplicative identity, respectively (we refrain from defining a notation for multiplicative inverses, as we do not need it). Furthermore, let z_F be the unique element in F such that $z_F + z_F = 1_F$. Note that for $F = \mathbb{Z}_p$ we get $1_F = 1$ and $z_F = \frac{p+1}{2}$.

Let $\mathcal{A}$ denote the set of operations $f(x, y) = z_F \cdot_F (x +_F y)$, where F is a finite field of prime order $p = |D|$ and $p > 2$. The proof will be partitioned into two main cases due to the following result.

Lemma 5.24 (see [26, 139])**.** *If* $\mathrm{Inv}(f)$ *is a maximal relational clone and* f *is a binary idempotent operation, then either*

1. $\mathrm{Inv}(f) = \mathrm{Inv}(g)$*, where* $g \in \mathcal{A}$*, or*
2. $B \in \mathrm{Inv}(f)$ *for some two-element* $B \subseteq D$*.*

The classification result is given in the next lemma together with a proof outline. Full proofs concerning the case when $\mathrm{Inv}(f) = \mathrm{Inv}(g)$ and $g \in \mathcal{A}$ can be found in Section 5.6.4.1. In Section 5.6.4.2 we study the case when f is a 2-semilattice operation. The result is a proof of Theorem 5.4, which is shown to imply a complete characterisation for domains D such that $|D| \leq 4$. Finally, in Section 5.6.4.3 we extend the classification to general domains under the assumption of Conjecture 5.2.

Lemma 5.25. *Let* f *be a binary commutative idempotent operation on* D *such that* $\mathrm{Inv}(f)$ *is a maximal relational clone, and let* Γ *be a constraint language such that* $\langle\Gamma\rangle = \mathrm{Inv}(f)$*.*

- *If* $\mathrm{Inv}(f) = \mathrm{Inv}(g)$ *for some* $g \in \mathcal{A}$*, then* W-MAX SOL(Γ) *is* **APX***-complete.*
- *Else if* $|D| \leq 4$ *and there exist* $a, b \in D$ *such that* $a < b$ *and* $f(a, b) = a$*, then let* a^* *be the minimal element (according to* $<$*) such that there is* $b^* \in D$ *which satisfies* $a^* < b^*$ *and* $f(a^*, b^*) = a^*$*. Then*
  - W-MAX SOL(Γ) *is* **poly-APX***-complete if* $a^* = 0$*, and*
  - **APX***-complete if* $a^* > 0$*.*
- *Otherwise, if* $|D| \leq 4$*, then* W-MAX SOL(Γ) *is in* **PO***.*

Proof. If $\mathrm{Inv}(f) = \mathrm{Inv}(g)$ and $g \in \mathcal{A}$, then the result follows from Section 5.6.4.1.

If there exist $a, b \in D$ such that $a < b$ and $f(a, b) = a$, then we need to consider two cases depending on a^*. If $a^* = 0$, then W-MAX SOL(Γ) is **poly-APX**-hard by Lemma 5.9 and a member of **poly-APX** by Lemma 5.7 since $\mathrm{CSP}(\Gamma \cup C_D)$ is in **P** [26, 20] (note that $C_D \subseteq \mathrm{Inv}(f)$ as f is idempotent). If $a^* > 0$, then W-MAX SOL(Γ) is **APX**-complete by Lemma 5.35 in Section 5.6.4.3 below.

Finally, if there do not exist any $a, b \in D$ such that $a < b$ and $f(a, b) = a$, then f acts as the max operation on every two-element $B \subseteq D$ such that $B \in \mathrm{Inv}(f)$. Lemma 5.36 shows that W-Max Sol(Γ) is in **PO**.

Lemma 5.35 and 5.36 assume that Conjecture 5.2 is true. As mentioned in Section 5.2 it was shown in [137] that it does hold when $|D| \leq 4$. □

5.6.4.1 f is Contained in $\mathcal{A}$

We will now prove that W-Max Sol(Γ) is **APX**-complete whenever $f \in \mathcal{A}$ and $\langle\Gamma\rangle = \mathrm{Inv}(f)$.

Lemma 5.26. *Let $f(x, y) = z_F \cdot_F (x +_F y)$, where F is a finite field of prime order $p = |D| > 2$ and $\mathrm{Inv}(f)$ is a maximal relational clone. Then,* W-Max Sol(Γ) *is* **APX***-complete if $\langle\Gamma\rangle = \mathrm{Inv}(f)$.*

Proof. We will give the proof for $F = \mathbb{Z}_p$ and after that we will argue that the proof can easily be adapted to the general case.

Let $q = \frac{p+1}{2}$, then f is the function $(x, y) \mapsto q(x + y) \pmod p$. We will show that $a(x, y, z) = x - y + z \pmod p$ is contained in the clone generated by f. Note that

$$\sum_{i=1}^{p-1} q^i = \frac{1 - q^p}{1 - q} - 1 = 0 \pmod p. \tag{5.3}$$

Here the first equality follows from the fact that we are summing the terms of a geometric progression. The second equality follows from Fermat's little theorem: $a^{p-1} = 1 \pmod p$ for any prime p and integer a not divisible by p. By using (5.3) and Fermat's little theorem again, we get

$$\sum_{i=1}^{p-2} q^i = -1 \pmod p. \tag{5.4}$$

Now,

$$\begin{aligned}
&\underbrace{f(f(f(\ldots f(f(f(}_{p-1 \text{ times}} x, z), y), y) \ldots), y), y) \\
&= q(q(q(\ldots q(q(q(x + z) + y) + y) + \cdots) + y) + y) \\
&= q^{p-1}x + q^{p-1}z + \sum_{i=1}^{p-2} q^i y \\
&= x - y + z \pmod p
\end{aligned}$$

where the final equality follows from (5.4) and Fermat's little theorem.

As any finite field F of prime order is isomorphic to $\mathbb{Z}_p$, it follows that $a(x, y, z) = x -_F y +_F z$ is contained in the clone generated by f for any such field. Since $\mathrm{Inv}(f)$ is a maximal relational clone, a is contained in the clone

generated by f, and a is not a projection, it follows that $\mathrm{Inv}(f) = \mathrm{Inv}(a)$. We now get containment in **APX** from Theorem 4.12 and **APX**-hardness from Theorem 5.23. □

5.6.4.2 f is a 2-semilattice Operation

In this section we will prove Theorem 5.4. That is, we will classify the complexity of W-MAX SOL(Γ) when $\langle\Gamma\rangle = \mathrm{Inv}(f)$ for all 2-semilattice operations f. It is noted in [21] that binary operations f such that $\mathrm{Inv}(f)$ is a maximal constraint language on $|D| \leq 4$ are either 2-semilattices or otherwise CSP(Γ) is **NP**-complete. Hence, we get a classification of the complexity of W-MAX SOL(Γ) when $\langle\Gamma\rangle = \mathrm{Inv}(f)$ is a maximal constraint language over $|D| \leq 4$ and f is a binary operation.

Lemma 5.27. *Let f be a 2-semilattice operation on D and let $\langle\Gamma\rangle = \mathrm{Inv}(f)$. If there exist $a, b \in D$ such that $a < b$, $f(a,b) = a$, and $a^* > 0$, where a^* is the minimal element such that there is b^* with $a^* < b^*$ and $f(a^*, b^*) = a^*$, then* W-MAX SOL(Γ) *is* **APX***-complete.*

Proof. The **APX**-hardness part is clear. What remains is to show that the problem is in **APX**. We can assume, without loss of generality, that $a = a^*$ and $b = b^*$. We begin by proving that $U = D \setminus \{0\}$ is in $\mathrm{Inv}(f)$. Assume that $f(x,y) = 0$ and $x, y > 0$; then $f(x, f(x,y)) = f(x,y) = 0$ and, consequently, $f(x,0) = 0$, contradicting the assumption that $a > 0$ was the minimal such element. Hence, $f(x,y) = 0$ if and only if $x = y = 0$. In particular U is in $\mathrm{Inv}(f)$.

We continue with the actual proof of the lemma. Let $I = (V, D, C, w)$ be an arbitrary instance of W-MAX SOL(Γ). Define $V' \subseteq V$ such that

$$V' = \{v \in V \mid s(v) = 0 \text{ for every solution } s \text{ to } I\}.$$

We see that V' can be computed in polynomial time: a variable v is in V' if and only if the CSP instance $(V, D, C \cup \{((v), U)\})$ is not satisfiable.

Given two assignments $\alpha, \beta : V \to D$, we define the assignment $f(\alpha, \beta)$ such that $f(\alpha,\beta)(v) = f(\alpha(v), \beta(v))$. We note that if α and β are solutions of I, then $f(\alpha,\beta)$ is a solution to I, too: indeed, arbitrarily choose one constraint $((x_1, \ldots, x_k), R) \in C$. Then, $(\alpha(x_1), \ldots, \alpha(x_k)) \in R$ and $(\beta(x_1), \ldots, \beta(x_k)) \in R$, which implies that

$$(f(\alpha(x_1), \beta(x_1)), \ldots, f(\alpha(x_k), \beta(x_k))) \in R,$$

too. Let $s_1, \ldots, s_m$ be an enumeration of all solutions of I and define

$$s^+ = f(s_1, f(s_2, f(s_3 \ldots f(s_{m-1}, s_m) \ldots))).$$

By the choice of V' and the fact that $f(c,d) = 0$ if and only if $c = d = 0$, we see that the solution s^+ has the following property: $s^+(v) = 0$ if and only if

$v \in V'$. Let p denote the second least element in D, and note that

$$\sum_{v \in V \setminus V'} w(v) \max D \geq \text{OPT}(I) \geq \sum_{v \in V \setminus V'} w(v)p = c.$$

Thus, by finding a solution with measure $\geq c$, we have approximated I within $(\max D)/p$, and W-MAX SOL(Γ) is in **APX**. To find such a solution, we consider the instance $I' = (V, D, C', w)$, where $C' = C \cup \{((v), U) \mid v \in V \setminus V'\}$. This instance has feasible solutions (since s^+ is a solution), and every solution has measure $\geq c$. Finally, a concrete solution can be found in polynomial time by the result in [32]. □

Lemma 5.28. *If f is a 2-semilattice operation such that $f \in \text{Pol}(\Gamma)$ and for all $a, b \in D$ such that $f(a, b) \in \{a, b\}$ we have $f(a, b) = \max\{a, b\}$, then* W-MAX SOL(Γ) *is in* **PO**.

Proof. What we will prove is that if f acts as max on all two-element $B \in \text{Inv}(f)$, then f is a generalised max operation and consequently W-MAX SOL(Γ) is in **PO**.

First note that if $a \neq b$ and $f(a, b) = a$, then by assumption $a > b$ and $f(a, b) > \min\{a, b\}$. Now, if $f(a, b) \neq a$, then $f(a, f(a, b)) = f(a, b)$ and by assumption f is max on $\{a, f(a, b)\}$, so $a < f(a, b)$ and thus $f(a, b) > \min\{a, b\}$. Now, $f(a, b) > \min\{a, b\}$ for all $a \neq b$. Moreover, f is idempotent, so f is a generalised max operation and tractability follows from Corollary 5.16. □

We are now ready to prove Theorem 5.4.

Proof (Of Theorem 5.4). The **PO** case follows from Lemma 5.28. The **APX**-complete case follows from Lemma 5.27 and, finally, the **poly-APX**-complete case is a consequence of Lemma 5.9. □

As mentioned in the first paragraph of this section, for any binary operation f on a domain of size at most four such that $\text{Inv}(f)$ is a maximal constraint language it is known that either f is a 2-semilattice operation or $\text{CSP}(\text{Inv}(f))$ is **NP**-complete. Hence, we have now completely classified the complexity of W-MAX SOL(Γ) for all constraint languages Γ such that $\langle\Gamma\rangle$ is maximal and $|D| \leq 4$.

5.6.4.3 Complete Classification under a Conjecture

In this section we will prove that the validity of Conjecture 5.2 implies a complete complexity classification of W-MAX SOL(Γ) for all constraint languages Γ such that $\langle\Gamma\rangle$ is maximal. Our proof builds on a construction that facilitates the study of operation f—the details are collected in Lemma 5.29. The underlying idea and the proof of Lemma 5.29 are inspired by Lemma 3 in [26]. Let f be a binary operation on D and define operations $f_1, f_2, \ldots : D^2 \to D$ inductively:

$$f_1(x, y) = f(x, y),$$
$$f_{n+1}(x, y) = f(x, f_n(x, y)).$$

These operations will be used intensively in this section.

Lemma 5.29. *Assume f to be a binary commutative idempotent operation on D such that* $\mathrm{Inv}(f)$ *is a maximal relational clone and* $\mathrm{Inv}(f) \neq \mathrm{Inv}(g)$ *for every $g \in \mathcal{A}$. The following hold:*

1. $f|_B = f_n|_B$ *for every $n \geq 1$ and every two-element $B \subseteq D$ in* $\mathrm{Inv}(f)$*; and*
2. $\mathrm{Inv}(f) = \mathrm{Inv}(f_n)$*, $n \geq 1$.*

Proof. We start with proving the first part of the lemma. Arbitrarily choose a two-element $\{a, b\} = B \subseteq D$ in $\mathrm{Inv}(f)$. There are two possible binary commutative idempotent operations on B, namely, max and min. We assume without loss of generality that $f|_B = \max$ and prove the result by induction over n. Since $f_1 = f$, the claim holds for $n = 1$. Assume it holds for $n = k$ and consider f_{k+1}. We see that $f_{k+1}(a, b) = f(a, f_k(a, b))$ and, by the induction hypothesis, $f_k(a, b) = \max(a, b)$. Hence, $f_{k+1}(a, b) = \max(a, \max(a, b)) = \max(a, b)$.

As for the second part of the lemma, obviously $f_n \in \mathrm{Pol}(\mathrm{Inv}(f))$ and thus $\mathrm{Inv}(f) \subseteq \mathrm{Inv}(f_n) \subseteq R_D$. Since $\mathrm{Inv}(f) \neq \mathrm{Inv}(g)$ for every $g \in \mathcal{A}$, we know from Lemma 5.24 that there is some two-element $B \in \mathrm{Inv}(f)$. By the proof above, we also know that $f|_B = f_n|_B$ so $f_n|_B$ (and consequently f_n) is not a projection. Thus, $\mathrm{Inv}(f_n) \neq R_D$, since $\mathrm{Inv}(f') = R_D$ if and only if f' is a projection. By the assumption that $\mathrm{Inv}(f)$ is a maximal relational clone and the fact that $\mathrm{Inv}(f) \subseteq \mathrm{Inv}(f_n) \subsetneq R_D$, we can draw the conclusion that $\mathrm{Inv}(f) = \mathrm{Inv}(f_n)$. □

We will now present some technical machinery that is needed for proving Lemmas 5.35 and 5.36. Recall the definition of fixity and the operation $\mathcal{F}$ from Section 5.2.

Lemma 5.30 (See [137, Lemma 28])**.** *Let f be an idempotent binary operation and let $n \in \mathbb{N}$. Then, $\mathcal{F}(f) \subseteq \mathcal{F}(f_n)$.*

Proof. Let $(x, y) \in \mathcal{F}(f)$. Then, either $f(x, y) = x$ or $f(x, y) = y$. Now

$$f(x,y) = x \Rightarrow f_n(x,y) = \underbrace{f(x, f(x, \ldots, f(x,y)\ldots))}_{n \text{ times}}$$

$$= \underbrace{f(x, f(x, \ldots, f(x,x)\ldots))}_{n-1 \text{ times}} = x \Rightarrow f_n(x,y) = x,$$

while

$$f(x,y) = y \Rightarrow f_n(x,y) = \underbrace{f(x, f(x, \ldots, f(x,y)\ldots))}_{n \text{ times}}$$

$$= \underbrace{f(x, f(x, \ldots, f(x,y)\ldots))}_{n-1 \text{ times}} \Rightarrow f_n(x,y) = y.$$

□

Assuming that Conjecture 5.2 holds, we get the following corollary as a consequence of Lemma 5.30.

Corollary 5.31. *Assuming Conjecture 5.2, if* $\mathrm{Inv}(f)$ *is a maximal relational clone such that* f *is a binary commutative idempotent operation and* $(x, y) \in \mathcal{F}(f_k)$ *(i.e.,* $f_k(x, y) \in \{x, y\}$*), then* $\{x, y\} \in \mathrm{Inv}(f)$*.*

Proof. By Lemma 5.30, we have $\mathcal{F}(f) \subseteq \mathcal{F}(f_k)$, and if Conjecture 5.2 holds, then $|\mathcal{F}(f)| = |\mathcal{F}(f_k)|$, which implies that $\mathcal{F}(f) = \mathcal{F}(f_k)$. Now, if $f_k(x, y) \in \{x, y\}$, then $f(x, y) \in \{x, y\}$, and by the commutativity of f we have $f(y, x) \in \{x, y\}$. Since f is idempotent, it is clear that $\{x, y\} \in \mathrm{Inv}(f)$. □

We continue by introducing a digraph associated with the binary operation f. This digraph enables us to make efficient use of Lemma 5.29. Given a binary operation $f : D^2 \to D$, we define $G_f = (V, E)$ such that $V = D \times D$ and $E = \{((a, b), (a, f(a, b))) \mid a, b \in D\}$. We make the following observations about G_f:

(1) an edge $((a, b), (a, c))$ implies that $f(a, b) = c$;

(2) every vertex has out-degree 1; and

(3) there is no edge $((a, b), (c, d))$ with $a \neq c$.

We extract some more information about G_f in the next three lemmas.

Lemma 5.32. *Assuming Conjecture 5.2, if* $\mathrm{Inv}(f)$ *is a maximal relational clone such that* f *is a binary commutative idempotent operation, then the digraph* G_f *contains no directed cycle.*

Proof. Assume G_f contains a directed cycle. Fact (3) allows us to assume (without loss of generality) that the cycle is $(0, 1), (0, 2), \ldots, (0, k), (0, 1)$ for some $k \geq 2$. Fact (1) tells us that $f(0, 1) = 2$, $f(0, 2) = 3, \ldots, f(0, k-1) = k$, and $f(0, k) = 1$. Furthermore, one can see that $f_2(0, 1) = 3$, $f_2(0, 2) = 4, \ldots$, and inductively $f_p(0, i) = i + p \pmod{k}$. This implies that $f_k(0, 1) = 1 + k \pmod{k} = 1$. By Corollary 5.31, $\{0, 1\}$ is a subalgebra of $\mathrm{Inv}(f)$, which contradicts the fact that $f(0, 1) = 2$. □

Lemma 5.33. *Assuming Conjecture 5.2, if* $\mathrm{Inv}(f)$ *is a maximal relational clone such that* f *is a binary commutative idempotent operation, then every path in* G_f *of length* $n \geq |D|$ *ends in a reflexive vertex; i.e.,* $f_n(a, b) = c$ *implies that* (a, c) *is a reflexive vertex.*

Proof. Assume that G_f contains a path P of length $n \geq |D|$. This path can contain at most $|D|$ distinct vertices by fact (3). Fact (2) together with the acyclicity (Lemma 5.32) of G_f implies that at least one vertex v on P is reflexive; by using fact (2) once again, we see that there exists exactly one reflexive vertex on P and that it must be the last vertex. □

Lemma 5.34. *Assuming Conjecture 5.2, if $f_n(a,b) = c$, then (a,c) is a reflexive vertex in G_f.*

Proof. By Lemma 5.33, every path in G_f of length $n \geq |D|$ ends in a reflexive vertex. Hence, $(a, f_n(a,b)) = (a,c)$ is a reflexive vertex. □

We will now use the information about G_f and prove the results we are aiming for.

Lemma 5.35. *Assume Conjecture 5.2 and let f be a binary commutative idempotent operation on D such that Γ is a maximal constraint language satisfying $\langle\Gamma\rangle = \mathrm{Inv}(f)$. If there exist $a, b \in D$ such that $a < b$, $f(a,b) = a$, and $a^* > 0$, where a^* is the minimal element such that there is b^* with $a^* < b^*$ and $f(a^*, b^*) = a^*$, then* W-MAX SOL(Γ) *is* **APX***-complete.*

Proof. We can assume, without loss of generality, that $a = a^*$ and $b = b^*$. The **APX**-hardness part follows from Lemma 5.8. What remains is to show that the problem is in **APX**. We begin the proof by proving that the unary relation $U = D \setminus \{0\}$ is a member of $\mathrm{Inv}(f)$. Consider the digraph G_f. As we have already observed in Lemma 5.32, there are no cycles $(a,b_1), \ldots, (a,b_k), (a,b_1)$, $k \geq 2$, in G_f, and every path of length $n \geq |D|$ ends in a reflexive vertex (by Lemma 5.33). Obviously, no vertex $(a,0)$ $(a > 0)$ in G_f is reflexive since this implies that $f(a,0) = 0$ which is a contradiction. In particular, there exists no path in G_f of length $n \geq |D|$ starting in a vertex (a,b) $(a > 0)$ and ending in a vertex $(a,0)$, since this implies that $(a,0)$ is reflexive by Lemma 5.33.

We can now conclude that $f_n(a,b) > 0$ when $a > 0$: if $f_n(a,b) = 0$, then $(a,0)$ is reflexive by Lemma 5.34, which would lead to a contradiction. Hence, $f_n(a,b) > 0$ whenever $a, b \in D \setminus \{0\} = U$ so U is in $\mathrm{Inv}(f_n)$, and, by Lemma 5.29(2), U is in $\mathrm{Inv}(f)$, too. We also note that $U \in \mathrm{Inv}(f)$ together with the assumption that $f(0,b) > 0$ for all $b > 0$ implies that $f(c,d) = 0$ if and only if $c = d = 0$. The rest of the proof is identical to the second part of the proof of Lemma 5.27. □

Lemma 5.36. *Assuming Conjecture 5.2, if f is a binary commutative idempotent operation such that $f \notin \mathcal{A}$, Γ is a maximal constraint language satisfying $\langle\Gamma\rangle = \mathrm{Inv}(f)$, and for all two-element $B \in \mathrm{Inv}(f)$ the operation f acts as the* max *operation on B, then* W-MAX SOL(Γ) *is in* **PO***.*

Proof. What we will prove is that if $f \notin \mathcal{A}$ and f acts as max on all two-element $B \in \mathrm{Inv}(f)$, then there exists a generalised max function f' such that $\mathrm{Inv}(f') = \mathrm{Inv}(f)$ and, hence, W-MAX SOL(Γ) is in **PO**.

Recall that there are no cycles in G_f and every path of length $n = |D|$ must end in a reflexive vertex by Lemmas 5.32 and 5.33. We also note that if G_f contains a reflexive vertex (a,c) with $a \neq c$, then $f(a,c) = c$ and $c > a$ since f is assumed to act as the max operation on all two-element $B \in \mathrm{Inv}(f)$.

We now claim that f_n is a generalised max operation. Arbitrarily choose $a, b \in D$. If $f_n(a,b) = c$ $(a \neq c)$, then there is a path in G_f from (a,b) to

a reflexive vertex (a, c), and $c > a$ as explained above. If $f_n(a, b) = a$, then $\{a, b\} \in \mathrm{Inv}(f)$ by Corollary 5.31. Since $f(a, b) = \max(a, b)$, Lemma 5.29(1) implies that $f_n(a, b) = \max(a, b)$. Thus, f_n is a generalised max-operation. □

5.7 Proof of Theorem 5.3

Proof (Of Theorem 5.3). We first prove the theorem for the case when $|D| \leq 4$ and Conjecture 5.2 is not assumed to hold. Part 1 follows from Corollary 5.16 (for generalised max-closed operations) and Lemma 5.19 (for constant operations).

The first part of Part 2 follows from Theorem 5.22 (containment in **APX** for affine operations), Theorem 5.23 (**APX**-hardness for affine operations) and Lemma 5.19 (constant operations). Now consider the case of a binary commutative operation. As $|D| \leq 4$, Part 2 follows from Lemma 5.25. Finally, the result for majority operations follows from Lemma 5.20.

Part 3 follows from Lemma 5.25 (binary commutative operation) and Lemma 5.20 (majority operation). Part 4 follows from Lemma 5.19 and Part 5 from Theorem 5.1.

This concludes the proof of Theorem 5.3 for the case when $|D| \leq 4$. We will now show that the theorem holds for arbitrary domains under the assumption that Conjecture 5.2 holds.

In all of the parts above, it is only for the binary commutative operations we need the restriction that $|D| \leq 4$. Hence, we only need to deal with the binary commutative operations under the assumption that Conjecture 5.2 holds.

Let f be a binary commutative operation. If $f \in \mathcal{A}$, then by Theorem 5.22 (containment in **APX** for affine operations) and Theorem 5.23 (**APX**-hardness for affine operations) the problem is **APX**-complete. This is one part of Part 2.

Otherwise $f \notin \mathcal{A}$ and by Lemma 5.24 there is some two-element subset B of the domain which is preserved by f. If f acts as a max-operation on every such two-element subset, f is, by Lemma 5.36, a generalised max operation and thus by Corollary 5.16 it is in **PO**. In this case Part 1 is applicable.

If there is some two-element B on which f acts as a min-operation but $f(0, b) > 0$ for every $b \in D$ or if $0 \notin D$, then by Lemma 5.35 the problem is **APX**-complete. In this case Part 2 is applicable.

Finally, if there is some $b \in D$ such that $f(0, b) = 0$, then by Lemma 5.7 and Lemma 5.9 the problem is **poly-APX**-complete. In this case Part 3 is applicable. □

Chapter 6

Bounded Occurrence Max Ones

6.1 Introduction

Recall that Max Sol is called Max Ones when the domain is $\{0,1\}$. In this chapter the approximability of W-Max Ones is studied when the number of variable occurrences is bounded by a constant. For conservative constraint languages we give a complete classification theorem when the number of occurrences is three or more and a partial classification when the bound is two. For the non-conservative case we prove that the problem is either trivial or equivalent to the corresponding conservative problem under polynomial-time many-one reductions.

For a positive integer k we use (W-)Max Ones(Γ)-k to denote the problem (W-)Max Ones(Γ) with the additional restriction that each variable occurs at most k times. Max Ones(Γ)-k can represent many well-known problems. As a first example, let R be the relation $\{(0,0),(1,0),(0,1)\}$. For $k \geq 3$ Independent Set in graphs of maximum degree k is precisely Max Ones($\{R\}$)-k. However, the more interesting case is perhaps $k = 2$ due to its connection to matching problems. Ordinary weighted maximum matching in graphs is, for example, straightforward to formulate with the constraint language

$$\Gamma_{\text{match}} = \{R_n \mid n \in \mathbb{N}\}$$

where R_n is the n-ary relation $\{\boldsymbol{x} \in \{0,1\}^n \mid |\boldsymbol{x}| \leq 1\}$. (For a boolean tuple $\boldsymbol{x}$ we use $|\boldsymbol{x}|$ to denote the number of ones in $\boldsymbol{x}$.) Given a graph $G = (V, E)$ we can construct an instance I of Max Ones(Γ_{match})-2 by letting the variables be E and for every vertex v of degree $d(v)$ in G we add a constraint $R_{d(v)}$ to I with scope $\{e \in E \mid v \in e\}$ (that is, the edges adjacent to v). The order of the variables in the constraint application does not matter as $R_{d(v)}$ is symmetric under permutation of the indices.

In the *b-factor problem* one is given a graph and is asked to select a maximum cardinality subset of the edges such that at most b selected edges are adjacent to any vertex. With $b = 1$ we get the ordinary matching problem. The b-factor problem can also be seen as a MAX ONES(Γ)-2 problem with the constraint language $\{R_n^b \mid n \in \mathbb{N}\}$ where R_n^b is the n-ary relation $\{\boldsymbol{x} \in \{0,1\}^n \mid |\boldsymbol{x}| \leq b\}$. Some other matching problems in graphs, such as the general factor problem [30, 41], can also be represented as a MAX ONES(Γ)-2 problem. The general factor problem corresponds to constraint languages which contains a subset of the relations R_n^I defined by

$$R_n^I = \{\boldsymbol{x} \in \{0,1\}^n \mid |\boldsymbol{x}| \in I\}$$

for each positive integer n and subset I of $[n]$.

The conservative bounded occurrence variant of CSP(Γ) has been studied by a number of authors [47, 56, 57, 83]. One result of that research is that the difficult case to classify is when the number of variable occurrences are restricted to two, in all other cases the bounded occurrence problem is no easier than the unrestricted problem. That is, CSP(Γ)-k is as hard as CSP(Γ) when $k \geq 3$ and Γ is conservative. For the $k = 2$ case the theory of Δ-matroids plays an important role. These objects will play an important role in our results as well; we refer the reader to Section 6.4 for more details. Kratochvíl et al. [106] have studied k-SAT-l, i.e., satisfiability where every clause have length k and there are at most l occurrences of each variable. k-SAT-l is a *non-conservative* constraint satisfaction problem. The complexity classification seems to be significantly harder for such problems compared to the conservative ones. In particular, Kratochvíl et al. [106] proves that there is a function f such that k-SAT-l is trivial if $l \leq f(k)$ (every instance has a solution) and **NP**-complete if $l \geq f(k) + 1$. Some bounds of f are given in [106], but the exact behaviour of f is unknown. For W-MAX ONES(Γ)-k it is possible to get around most of the problems with non-conservative constraint languages as one can simulate the unary constraint $\{1\}$ with large weights.

This chapter is organised as follows: in Section 6.2 we define some notation and present some tools which are specific to this chapter. Section 6.3 and 6.4 contains the results for three or more occurrences and two occurrences, respectively. Section 6.5 contains the results for the nonconservative case.

6.2 Preliminaries

The *Hamming distance* between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ will be denoted by $d_H(\boldsymbol{x}, \boldsymbol{y})$ and is defined to be the number of coordinates in which $\boldsymbol{x}$ differs from $\boldsymbol{y}$. As mentioned in the introduction we use the notation $|\boldsymbol{t}|$ to denote the number of ones in the tuple $\boldsymbol{t}$. Unless explicitly stated otherwise we assume that the constraint languages we are working with are *conservative*,

i.e., every unary relation is a member of the constraint language (in the boolean domain, which we are working with, this means that $\{(0)\}$ and $\{(1)\}$ are in the constraint language). We define the following relations

- $\text{NAND}^m = \{(x_1, \ldots, x_m) \mid x_1 + \ldots + x_m < m\}$,
- $EQ^m = \{(x_1, \ldots, x_m) \mid x_1 = x_2 = \ldots = x_m\}$,
- $\text{IMPL} = \{(0,0), (0,1), (1,1)\}$,
- $c_0 = \{(0)\}$,
- $c_1 = \{(1)\}$,

and the function

$$h_n(x_1, x_2, \ldots, x_{n+1}) = \bigvee_{i=1}^{n+1} (x_1 \wedge \ldots \wedge x_{i-1} \wedge x_{i+1} \wedge \ldots \wedge x_{n+1}).$$

This function can also be defined as $h_n(x_1, \ldots, x_{n+1}) = 1$ if $\sum_{i=1}^{n+1} x_i \geq n$ and $h_n(x_1, \ldots, x_{n+1}) = 0$ otherwise.

For a tuple $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and a set of coordinates $A \subseteq [n]$, $\boldsymbol{x} \oplus A$ is defined to be the tuple $(y_1, y_2, \ldots, y_n)$ where $y_i = x_i$ if $i \notin A$ and $y_i = 1 - x_i$ otherwise. We extend this notation to relations: if $R \subseteq \{0,1\}^n$ and $A \subseteq [n]$ then $R \oplus A = \{\boldsymbol{t} \oplus A \mid \boldsymbol{t} \in R\}$. By abuse of notation we will also write $\boldsymbol{x} \oplus k$ and $R \oplus k$ for an integer $k \in [n]$ to denote $\boldsymbol{x} \oplus \{k\}$ and $R \oplus \{k\}$, respectively.

We need a variant of pp-definitions which is a bit stronger than the one presented in Chapter 1, because we have to be careful with how many occurrences we use of each variable.

Definition 6.1 (k-representable)**.** *An n-ary relation R is* k-representable *by a set of relations F if there is a collection of constraints $C_1, \ldots, C_l$ with constraint relations from F over variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ (called* primary variables*) and $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ (called* auxiliary variables*) such that,*

- *the primary variables occur at most once in the constraints;*
- *the auxiliary variables occur at most k times in the constraints; and*
- *for every tuple $\boldsymbol{z}$, $\boldsymbol{z} \in R$ if and only if there is an assignment to $\boldsymbol{y}$ such that $\boldsymbol{x} = \boldsymbol{z}$ satisfies all of the constraints $C_1, C_2, \ldots, C_l$.*

The intuition behind the definition is that if every relation in Γ_1 is k-representable by relations in Γ_2 then W-Max Ones(Γ_2)-k is no easier than W-Max Ones(Γ_1)-k. This is formalised in Lemma 6.2 below.

Lemma 6.2. *For constraint languages Γ_1 and Γ_2 if every relation in Γ_1 can be k-represented by Γ_2 then* W-Max Ones(Γ_1)-$k \leq_S$ W-Max Ones(Γ_2)-k.

Proof. Given an instance $I = (V, C, w)$ of W-MAX ONES(Γ_1)-k, we will construct an instance $I' = (V', C', w')$ of W-MAX ONES(Γ_2)-k, in polynomial time. For each $c \in C$, add the k-representation of c to C' and also add all variables which participate in the representation to V' in such a way that the auxiliary variables used in the representation are distinct from all other variables in V'. Let $w'(x) = w(x)$ for all $x \in V$ and $w(x) = 0$ if $x \notin V$ (i.e., all auxiliary variables will have weight zero).

It is not hard to see that: (a) every variable in I' occurs at most k times (b) $\text{OPT}(I') = \text{OPT}(I)$, and (c) given a solution s' to I' we can easily construct a solution s to I (let $s(x) = s'(x)$ for every $x \in V$) such that $m(I, s) = m(I', s')$. Hence, there is an S-reduction from W-MAX ONES(Γ_1)-k to W-MAX ONES(Γ_2)-k. □

We will do several reductions from MAXIMUM INDEPENDENT SET (hereafter denoted by MIS) which is **poly-APX**-complete [101]. We will also use the fact that for any $k \geq 3$, MIS restricted to graphs of degree at most k is **APX**-complete [122]. We will denote the latter problem by MIS-k.

6.2.1 Co-clones and Polymorphisms

For a set of relations Γ we define a closure operator $\langle\Gamma\rangle_{\neq}$ as the set of relations that can be expressed with relations from Γ using existential quantification and conjunction (note that we are only allowed to use the relations in Γ, hence equality is not necessarily allowed). The operator $\langle\cdot\rangle_{\neq}$ is different from the usual closure operator $\langle\cdot\rangle$ as equalities cannot be used whenever one wants in $\langle\cdot\rangle_{\neq}$. However, note that $\langle\Gamma \cup \{EQ^2\}\rangle_{\neq} = \langle\Gamma\rangle$ for any constraint language Γ.

We say that a set of relations B is a *plain basis* for a constraint language Γ if every relation in Γ can be expressed with relations from B using relations from $B \cup \{EQ^2\}$ and conjunction. Note that this differs from the definition of the closure operator $\langle\cdot\rangle$ as we do not allow existential quantification. We refer the reader to [45] for more information on plain bases.

We can not only study the co-clones when we try to classify MAX ONES(Γ)-k because the complexity of the problem does not only depend on the co-clone $\langle\Gamma\rangle$. However, the co-clone lattice with the corresponding plain bases and invariant functions will help us in our classification effort. Furthermore, as we mostly study the conservative constraint languages we can concentrate on the co-clones which contain c_0 and c_1. Figure 6.1 contains the conservative part of Post's lattice and Table 6.1 contains the plain bases for the relational clones which will be interesting to us (co-clones at and below IV_2 have been omitted as W-MAX ONES is in **PO** there).

6.3 Three or More Occurrences

In this section we will prove a classification theorem for W-MAX ONES(Γ)-k where $k \geq 3$. The main result of this section is the following theorem.

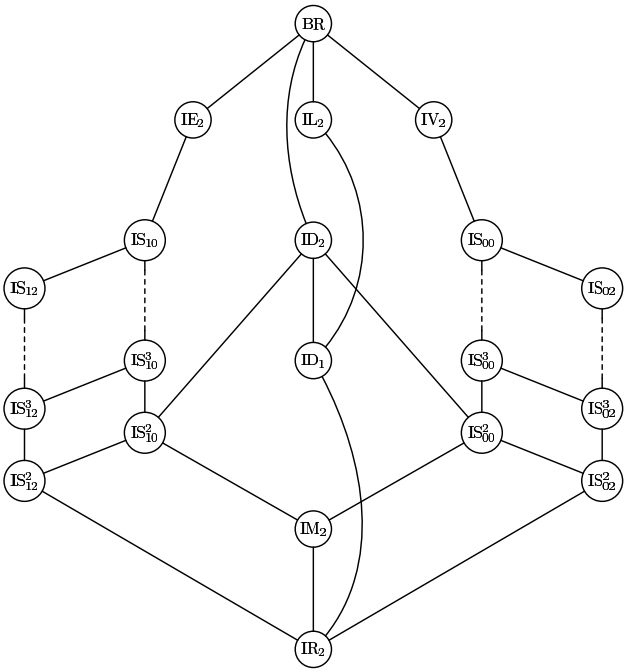


Figure 6.1: Lattice of conservative co-clones.

Co-clone	Base for clone	Plain basis
IE_2	and	$\{\text{NAND}^k \mid k \in \mathbb{N}\} \cup$ $\{(\neg x_1 \vee \ldots \vee \neg x_k \vee y) \mid k \in \mathbb{N}\}$
IS_{10}	$x \wedge (y \vee z)$	$\{c_1, \text{IMPL}\} \cup \{\text{NAND}^k \mid k \in \mathbb{N}\}$
IS^m_{10}	$x \wedge (y \vee z), h_m$	$\{c_1, \text{IMPL}, \text{NAND}^m\}$
IS_{12}	$x \wedge (y \vee \neg z)$	$\{EQ^2, c_1\} \cup \{\text{NAND}^k \mid k \in \mathbb{N}\}$
IS^m_{12}	$x \wedge (y \vee \neg z), h_m$	$\{EQ^2, c_1, \text{NAND}^m\}$
IL_2	$x \oplus y \oplus z$	$\{x_1 \oplus \ldots \oplus x_k = c \mid k \in \mathbb{N}, c \in \{0,1\}\}$
ID_2	$xy \vee yz \vee xz$	$\{c_0,\ c_1,\ x \vee y,\ \text{IMPL},\ \text{NAND}^2\}$
ID_1	$xy \vee y(\neg z) \vee y(\neg z)$	$\{c_0,\ c_1,\ x \oplus y = 0,\ x \oplus y = 1\}$
IM_2	and, or	$\{c_0, c_1, \text{IMPL}\}$
IR_2	$or, x \wedge (y \oplus z \oplus 1)$	$\{EQ^2, c_0, c_1\}$

Table 6.1: Plain bases for some relational clones. The list of plain bases are from [45]. (In [45] the listed plain basis for IS^m_{12} is $\{EQ^2, c_1\} \cup \{\text{NAND}^k | k \leq m\}$ however, if we have NAND^m then NAND^{m-1} can be represented without auxiliary variables by $\text{NAND}^{m-1}(x_1, x_2, \ldots, x_{m-1}) \iff \text{NAND}^m(x_1, x_1, x_2, x_3, \ldots, x_{m-1})$, hence the set of relations listed in the table above is a plain basis for IS^m_{12}. The same modification has been done to IS^m_{10}.)

Theorem 6.3. *Let Γ be a conservative constraint language and $k \geq 3$.*

1. *If $\Gamma \subseteq IV_2$, then* W-Max Ones(Γ)-k *is in* **PO**;
2. *else if $IS_{12}^2 \subseteq \langle\Gamma\rangle \subseteq IS_{12}$, then* W-Max Ones($\Gamma$)-$k$ *is* **APX**-*complete if EQ^2 is not k-representable by Γ and* W-Max Ones(Γ)-k *is* **poly-APX**-*complete otherwise;*
3. *otherwise,* W-Max Ones(Γ) *and* W-Max Ones(Γ)-k *are equivalent under S-reductions.*

The first part of Theorem 6.3 follows from Khanna et al.'s results for Max Ones [102]. Intuitively the second part follows from the fact that W-Max Ones($\{\text{NAND}^2\}$) is equivalent to MIS, hence if we have access to the equality relation then the problem gets **poly-APX**-complete (when $k \geq 3$ and we have access to equality, it is not hard to see that one can simulate any number of variable occurrences). On the other hand, if we do not have the equality relation then we essentially get MIS-k, for some k, which is **APX**-complete. We give the proofs for the second part in Section 6.3.1.

The third part is dealt with in Section 6.3.2 and finally in Section 6.3.3 we assemble the pieces and prove Theorem 6.3.

6.3.1 The Second Part: $IS_{12}^2 \subseteq \langle\Gamma\rangle \subseteq IS_{12}$

In this section we will prove a number of lemmas which will allow us to prove the second part of Theorem 6.3. The following lemma will be used in several places to prove hardness results.

Lemma 6.4. *Let Γ be a constraint language such that $\text{Pol}(\Gamma) = \text{Pol}(IS_{1\alpha}^m)$ for some integer m and $\alpha \in \{0, 2\}$, then $NAND^m$ can be 2-represented by Γ.*

Proof. As $\text{Pol}(\Gamma) = \text{Pol}(IS_{1\alpha}^m)$, Γ is invariant under h_m and not invariant under h_{m-1}. Hence, there is some $R \in \Gamma$ which is not invariant under h_{m-1}. Let r be the arity of R and let $X \subseteq [r]$ be a set of minimal cardinality such that there exist tuples $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m} \in \text{pr}_X\ R$ which satisfy $h_{m-1}(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m}) = \boldsymbol{z} \notin \text{pr}_X\ R$. If there is a coordinate $i \in X$ such that $\boldsymbol{x_1}(i) = \boldsymbol{x_2}(i) = \ldots = \boldsymbol{x_m}(i)$ then $\boldsymbol{z}(i) = \boldsymbol{x_1}(i)$. If $\boldsymbol{z} \oplus i \notin \text{pr}_X\ R$, then X is not minimal as $\text{pr}_{X\setminus\{i\}}\ \boldsymbol{z} \notin \text{pr}_{X\setminus\{i\}}\ R$. Hence, as we have assumed X to be minimal we must have $\boldsymbol{z} \oplus i \in \text{pr}_X\ R$. However, this means that $h_m(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m}, \boldsymbol{z} \oplus i) = \boldsymbol{z} \notin \text{pr}_X\ R$ which is a contradiction with the assumption that R is invariant under h_m. We conclude that no coordinate is constant in every $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m}$.

Now assume that there is a coordinate $j \in X$ such that $\boldsymbol{z}(j) = 0$, then for X to be minimal we must have $\boldsymbol{z} \oplus j \in \text{pr}_X\ R$, by the argument above. However, $h_m(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m}, \boldsymbol{z} \oplus j) = \boldsymbol{z} \notin \text{pr}_X\ R$, a contradiction, hence there is no $j \in X$ such that $\boldsymbol{z}(j) = 0$.

We can assume that $|X| \geq m$ because every relation of arity less than m which is invariant under h_m is also invariant under h_{m-1} [28, Proposition 3.7].

We do now know three things, no coordinate in X is constant in every $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m}$, $\boldsymbol{z} = (1, 1, \ldots, 1)$, and $|X| \geq m$. As $\boldsymbol{z} = (1, 1, \ldots, 1)$ there is at most one zero for every given coordinate $i \in X$ among $\boldsymbol{x_1}(i)$, $\boldsymbol{x_2}(i), \ldots,$ $\boldsymbol{x_m}(i)$. Furthermore, as $\boldsymbol{z} = (1, 1, \ldots, 1) \notin R$ we must have at least one zero in every $\boldsymbol{x_1}$, $\boldsymbol{x_2}, \ldots, \boldsymbol{x_m}$.

Now assume that there are two distinct coordinates $i, j \in X$ such that $\boldsymbol{x_k}(i) = \boldsymbol{x_k}(j) = 0$ for some $k \in [m]$. Let $I = \{i' \in X \setminus \{j\} \mid \boldsymbol{x_k}(i') = 0\}$. If $\boldsymbol{x_k} \oplus I \in R$ we can replace $\boldsymbol{x_k}$ by $\boldsymbol{x_k} \oplus I$ and iterate the proof. On the other hand, if $\boldsymbol{x_k} \oplus I \notin R$, then X is not minimal as $X \setminus \{j\}$ is of smaller cardinality and $\mathrm{pr}_{X \setminus \{j\}} R$ is not closed under h_{m-1} either.

This implies that $\boldsymbol{x_i} = (1, 1, \ldots, 1) \oplus \pi(i)$ for some permutation π on $[|X|]$. It is not hard to see that by using the fact that R is invariant under *and* we can get any tuple $\boldsymbol{y} = (y_1, y_2, \ldots, y_{|X|})$ such that $y_1 + y_2 + \ldots + y_{|X|} < |X|$ by applying *and* to the $\boldsymbol{x_i}$:s an appropriate number of times. Hence, we must have $\mathrm{pr}_X\ R = \mathrm{NAND}^{|X|}$. Finally, as R is invariant under h_m this implies that $|X| \leq m$ and hence $|X| = m$. □

Lemma 6.5. *Let* Γ *be a constraint language. If* $\mathrm{Pol}(\Gamma) = \mathrm{Pol}(IS_{12}^m)$ *for some* $m \geq 2$ *and* $EQ^2 \notin \langle\Gamma\rangle_{\neq}$*, then* $\langle\Gamma \cup \{c_0, c_1\}\rangle_{\neq} = \langle\{NAND^m, c_1\}\rangle_{\neq}$.

Proof. We will denote NAND^m by N. Let $R \in \Gamma$ and let r be the arity of R. $B = \{N, EQ^2, c_1\}$ is a plain basis for IS_{12}^m (see Table 6.1) and hence also a plain basis for Γ. As B is a plain basis for Γ there is an implementation of R on the following form,

$$\begin{aligned} R(x_1, \ldots, x_r) \iff & N(x_{k_1^1}, x_{k_1^2}, \ldots, x_{k_1^m}) \wedge \ldots \wedge N(x_{k_n^1}, \ldots, x_{k_n^m}) \wedge \\ & EQ^2(x_{l_1^1}, x_{l_1^2}) \wedge \ldots \wedge EQ^2(x_{l_c^1}, x_{l_c^2}) \\ & c_1(x_{c_1}) \wedge \ldots \wedge c_1(x_{c_w}) \end{aligned}$$

for some n, c and w such that $k_i^j \in [r]$, $l_i^j \in [r]$ and $c_i \in [r]$.

We can assume that all equalities are of the form $EQ^2(x_i, x_j)$ for $i \neq j$. If there is such an equality there are a number of cases to consider,

1. $\mathrm{pr}_{\{i,j\}} R = \{(0, 0), (1, 1)\}$,
2. $\mathrm{pr}_{\{i,j\}} R = \{(1, 1)\}$, and
3. $\mathrm{pr}_{\{i,j\}} R = \{(0, 0)\}$.

We cannot have equalities of type 1 because then $EQ^2 \in \langle\{R\}\rangle_{\neq}$. Furthermore, equalities of type 2 and 3 can be replaced by constraints of the form $c_1(x_i) \wedge c_1(x_j)$ and $N(x_i, \ldots, x_i) \wedge N(x_j, \ldots, x_j)$, respectively.

The conclusion is that $R \in \langle\{N, c_1\}\rangle_{\neq}$ (so EQ^2 is not needed) and hence $\langle\Gamma \cup \{c_0, c_1\}\rangle_{\neq} \subseteq \langle\{N, c_1\}\rangle_{\neq}$. The other inclusion, $\langle\{N, c_1\}\rangle_{\neq} \subseteq \langle\{R, c_0, c_1\}\rangle_{\neq}$, is given by Lemma 6.4. □

We are now ready to prove the containment result.

Lemma 6.6. *Let* Γ *be a constraint language. If* $\Gamma \subseteq IS_{12}^m$ *for some* m *and* $EQ^2 \notin \langle\Gamma\rangle_{\neq}$*, then* W-MAX ONES($\Gamma$)-$k$ *is in* **APX**.

Proof. Lemma 6.5 tells us that $\langle\Gamma\rangle_{\neq} \subseteq \langle\{\text{NAND}^m, c_1\}\rangle_{\neq}$ so an instance J of W-MAX ONES(Γ)-k can be reduced to an instance J' of W-MAX ONES($\{\text{NAND}^m, c_1\}$)-k' for some constant k'. To prove the lemma it is therefore sufficient show that W-MAX ONES($\{\text{NAND}^m, c_1\}$)-l is in **APX** for every fixed l.

Let $I = (V, C, w)$ be an instance of W-MAX ONES($\{\text{NAND}^m, c_1\}$)-l, for an arbitrary l, and assume that $V = \{x_1, \ldots, x_n\}$. By Schaefer's result [130] we can decide in polynomial time whether I has a solution or not. Hence, we can safely assume that I has a solution. If a variable occurs in a constant constraint, say $c_1(x)$, then x must have the same value in every model of I. Thus, we can eliminate all such variables and assume that I only contains constraints of the type $\text{NAND}^m(x_1, \ldots, x_m)$.

We will give a polynomial-time algorithm that creates a satisfying assignment s to I with measure at least $\frac{1}{l+1}\text{OPT}(I)$. Hence we have an $\frac{1}{l+1}$-approximate algorithm proving that W-MAX ONES(IS_{12})-l is in **APX**.

The algorithm is as follows: repeatedly delete from I any variable x_i having maximum weight and all variables that appear together with x_i in a clause of size two. In s we assign 1 to x_i and 0 to all variables appearing together with x_i in a clause of size two.

For simplicity, assume that the algorithm chooses variables $x_1, x_2, \ldots, x_t$ before it stops. If the algorithm at some stage choose a variable x with weight $w(x)$, then, in the worst case, it is forced to set l (remember that no variable occurs more than l times in I) variables to 0 and each of these variables have weight $w(x)$. This implies that $(l+1) \cdot \sum_{i=1}^{t} w(x_i) \geq \sum_{i=1}^{n} w(x_i)$ and

$$m(I, s) = \sum_{i=1}^{t} w(x_i) \geq \frac{1}{l+1} \sum_{i=1}^{n} w(x_i) \geq \frac{\text{OPT}(I)}{l+1}.$$

□

Lemma 6.7. *Let* Γ *be a constraint language such that* $IS_{12}^2 \subseteq \langle\Gamma\rangle \subseteq IS_{12}$ *then* W-MAX ONES(Γ)-k *is* **APX**-*hard for* $k \geq 3$.

Proof. Note that MIS-3 is exactly the same as MAX ONES($\{\text{NAND}^2\}$)-3. The lemma then follows from the fact that MIS-3 is **APX**-hard, Lemma 6.4, and Lemma 6.2. □

6.3.2 The Third Part

In this section we will prove the third part of Theorem 6.3. Feder [56, Theorem 3, fact 1] has proved the following lemma (Dalmau and Ford gave another proof for the same result in [47, Appendix A]). Note that we only need to use 2-representations here. This will be useful to us in Section 6.4.

Lemma 6.8. *If there is a relation R in the constraint language Γ such that $R \notin IE_2$, then either $x \vee y$ or $x \neq y$ can be 2-represented by Γ. Similarly, if there is a relation $R \in \Gamma$ such that $R \notin IV_2$, then either $NAND^2$ or $x \neq y$ can be 2-represented.*

We can use the lemma above to get a 2-representation of either EQ^2 or IMPL. We will later, in Lemma 6.11, show that those relations make the problem as hard as the unbounded occurrence variant.

Lemma 6.9. *If there is a relation R in the constraint language Γ such that $R \notin IE_2$ and $R \notin IV_2$, then either EQ^2 or IMPL can be 2-represented by Γ.*

Proof. From Lemma 6.8 we know that either $x \neq y$ or both $x \vee y$ and NAND^2 are 2-representable. In the first case $\exists z : x \neq z \wedge z \neq y$ is a 2-representation of EQ^2. In the second case $\exists z : \mathrm{NAND}^2(x, z) \wedge (z \vee y)$ is a 2-representation of $\mathrm{IMPL}(x, y)$. □

To get the desired hardness results for the IS_{10} chain we need to prove that we can represent EQ^2 or IMPL in that case too. To this end we have the following lemma.

Lemma 6.10. *If there is a relation R in the constraint language Γ such that $R \in IE_2$ and $R \notin IS_{12}$, then either EQ^2 or IMPL can be 2-represented by Γ.*

Proof. Let r be the arity of R. Then, as $R \notin IS_{12}$, there exists a set of minimal cardinality $I \subseteq [r]$, such that $\mathrm{pr}_I\, R \notin IS_{12}$.

As $g(x, y) = x \wedge y$ is a base of the clone which corresponds to IE_2, $\mathrm{pr}_I\, R \in IE_2$ implies that g is a polymorphism of $\mathrm{pr}_I\, R$. Furthermore, as $f(x, y, z) = x \wedge (y \vee \neg z)$ is a base of the clone which corresponds to IS_{12}, $\mathrm{pr}_I\, R \notin IS_{12}$ implies that f is *not* a polymorphism of $\mathrm{pr}_I\, R$. Hence, there exists tuples $\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3} \in \mathrm{pr}_I\, R$ such that $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3}) = \boldsymbol{t} \notin \mathrm{pr}_I\, R$.

There exists a coordinate $l_1, 1 \leq l_1 \leq r$ such that $(\boldsymbol{t_1}(l_1), \boldsymbol{t_2}(l_1), \boldsymbol{t_3}(l_1)) = (1, 0, 1)$, because otherwise $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3}) = \boldsymbol{t_1}$ (see Table 6.2). Similarly there exists a coordinate $l_2, 1 \leq l_2 \leq r$ such that $(\boldsymbol{t_1}(l_2), \boldsymbol{t_2}(l_2), \boldsymbol{t_3}(l_2))$ is equal to one of $(0, 1, 0)$, $(0, 1, 1)$ or $(1, 0, 0)$, because otherwise $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3}) = \boldsymbol{t_2}$. From now on, the case $(\boldsymbol{t_1}(l_2), \boldsymbol{t_2}(l_2), \boldsymbol{t_3}(l_2)) = (1, 0, 0)$ will be denoted by (*). Finally, there also exists a coordinate $l_3, 1 \leq l_3 \leq r$ such that $(\boldsymbol{t_1}(l_3), \boldsymbol{t_2}(l_3), \boldsymbol{t_3}(l_3))$ is equal to one of $(0, 0, 1)$, $(0, 1, 1)$, $(1, 0, 0)$, $(1, 0, 1)$ or $(1, 1, 0)$, because otherwise $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3}) = \boldsymbol{t_3}$. The specific case when $(\boldsymbol{t_1}(l_3), \boldsymbol{t_2}(l_3), \boldsymbol{t_3}(l_3)) = (1, 0, 0)$ will be denoted by (**).

As $\mathrm{pr}_I\, R$ is invariant under g we can place additional restrictions on l_1, l_2 and l_3. In particular, there has to be coordinates l_1, l_2 and l_3 such that we have at least one of the cases (*) or (**), because otherwise $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3}) = g(\boldsymbol{t_1}, \boldsymbol{t_2})$, which is in $\mathrm{pr}_I\, R$ and we have assumed that $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3}) \notin \mathrm{pr}_I\, R$. There is no problem in letting $l_2 = l_3$ since we will then get both (*) and (**). This will be assumed from now on. We can also assume, without loss of generality, that $l_1 = 1$ and $l_2 = l_3 = 2$. We can then construct a

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Table 6.2: The function f in the proof of Lemma 6.10.

3-representation as

$$R_\phi(x,y) \iff \exists z_3 \ldots \exists z_r : \mathrm{pr}_I\, R(x,y,z_3,\ldots,z_r) \wedge c_{k_3}(z_3) \wedge c_{k_4}(z_4) \wedge \ldots \wedge c_{k_r}(z_r)$$

where $k_i = f(\boldsymbol{t_1}(i), \boldsymbol{t_2}(i), \boldsymbol{t_3}(i))$ for $3 \leq i \leq r$. We will now prove that R_ϕ is equal to one of the relations we are looking for.

If $(0,1) \in R_\phi$, then we would have $\boldsymbol{t} \in \mathrm{pr}_I\, R$, which is a contradiction, so $(0,1) \notin R_\phi$. We will now show that $(0,0) \in R_\phi$. Assume that $(0,0) \notin R_\phi$. Then, $R^* = \mathrm{pr}_{I \setminus \{l_2\}}\, R$ is not in IS_{12} which contradicts the minimality of I. To see this consider the following table of possible tuples in $\mathrm{pr}_I\, R$,

	$1 = l_1$	$2 = l_2 = l_3$	3	4	...
$\boldsymbol{t_1}$	1	1	$\boldsymbol{t_1}(3)$	$\boldsymbol{t_1}(4)$	...
$\boldsymbol{t_2}$	0	0	$\boldsymbol{t_2}(3)$	$\boldsymbol{t_2}(4)$	...
$\boldsymbol{t_3}$	1	0	$\boldsymbol{t_3}(3)$	$\boldsymbol{t_3}(4)$	...
$\boldsymbol{t}$	0	1	$f(\boldsymbol{t_1}(3), \boldsymbol{t_2}(3), \boldsymbol{t_3}(3))$	$f(\boldsymbol{t_1}(4), \boldsymbol{t_2}(4), \boldsymbol{t_3}(4))$	...
$\boldsymbol{a}$	0	0	$f(\boldsymbol{t_1}(3), \boldsymbol{t_2}(3), \boldsymbol{t_3}(3))$	$f(\boldsymbol{t_1}(4), \boldsymbol{t_2}(4), \boldsymbol{t_3}(4))$	...

We know that $\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3} \in \mathrm{pr}_I\, R$ and we also know that $\boldsymbol{t} \notin \mathrm{pr}_I\, R$. Furthermore, if $\boldsymbol{a} \notin \mathrm{pr}_I\, R$, then $\mathrm{pr}_{I \setminus \{l_2\}}\, f(\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3}) \notin \mathrm{pr}_I\, R$ which means that I is not minimal. The conclusion is that we must have $(0,0) \in R_\phi$. In the same way it is possible to prove that unless $(1,1) \in R_\phi$, I is not minimal.

To conclude, we have proved that $(0,0), (1,1) \in R_\phi$ and $(0,1) \notin R_\phi$, hence we either have $R_\phi = EQ^2$ or $R_\phi = \{(0,0),(1,0),(1,1)\}$. □

It is now time to use our implementations of EQ^2 or IMPL to prove hardness results.

Lemma 6.11. *If EQ^2 or IMPL is 3-representable by the constraint language Γ then* W-MAX ONES(Γ) $\leq_S$ W-MAX ONES(Γ)-3.

The idea of the proof is that as either EQ^2 or IMPL is available we can construct a cycle of constraints among variables and such a cycle forces every variable in the cycle to obtain the same value. Furthermore, each variable

occurs only twice in such a cycle so we have one occurrence left for each variable.

Proof. Let $I = (V, C, w)$ be an instance of W-Max Ones(Γ). We will start with the case when IMPL is 3-representable.

If IMPL is 3-representable we can reduce I to an instance $I' = (V', C', w')$ of W-Max Ones(Γ)-2 as follows: for each variable $v_i \in V$, let o_i be the number of occurrences of v_i in I, we introduce the variables $v_i^1, \ldots, v_i^{o_i}$ in V'. We let $w'(v_i^1) = w(v_i)$ and $w'(v_i^j) = 0$ for $j \neq 1$. We also introduce the constraints IMPL(v_i^k, v_i^{k+1}) for $k, 1 \leq k \leq o_i - 1$ and IMPL$(v_i^{o_i}, v_i^1)$ into C'. For every $i, 1 \leq i \leq |V|$ those constraints make the variables $v_i^1, \ldots, v_i^{o_i}$ have the same value in every feasible solution of I'.

For every constraint $c = (R, \boldsymbol{s}) \in C$ the constraint scope $\boldsymbol{s} = (v_{l_1}, \ldots, v_{l_m})$ is replaced by $\boldsymbol{s'} = (v_{l_1}^{k_1}, \ldots, v_{l_m}^{k_m})$ and $(R, \boldsymbol{s'})$ is added to C'. The numbers $k_1, \ldots, k_m$ are chosen in such a way that every variable in V' occur exactly three times in I'. This is possible since there are o_i variables in V' for every $v_i \in V$. It is clear that the procedure described above is an S-reduction from W-Max Ones(Γ) to W-Max Ones(Γ)-3.

We now consider the case when EQ^2 is 3-representable. The instance I can easily be S-reduced to an instance I' of W-Max Ones($\Gamma \cup \{EQ^2\}$)-3. And as EQ^2 is 3-representable by Γ we are done, as every constraint involving EQ^2 can be replaced by the 3-representation of EQ^2 and any auxiliary variables used in the representation can be assigned the weight zero. □

6.3.3 Putting the Pieces Together

We are now ready to give the proof of the classification theorem for three or more occurrences.

Proof (Of Theorem 6.3).

Part 1. Follows directly from Khanna et al.'s results for Max Ones [102].

Part 2. The **APX**-hardness follows from Lemma 6.7. Containment in **APX** follows from Lemma 6.6. If EQ^2 is k-representable by Γ then the result follows from Lemma 6.11 and Khanna et al.'s results for Max Ones [102].

Part 3. There are two possibilities, the first one is that $\Gamma \not\subseteq IE_2$ and $\Gamma \not\subseteq IV_2$, the second case is that $\Gamma \subseteq IE_2$ and $\Gamma \not\subseteq IS_{12}$.

In the first case we can use the 3-representation of EQ^2 or IMPL from Lemma 6.9. The result then follows from Lemma 6.11. In the second case the result follows from Lemma 6.10 and Lemma 6.11. □

6.4 Two Occurrences

In this section we study W-Max Ones(Γ)-2. We are not able to present a complete classification for this case but a partial classification is achieved. We completely classify the co-clones IL_2 and ID_2. For Γ such that $\Gamma \not\subseteq IL_2, ID_2$ we show that if there is a relation which is not a Δ-matroid relation

(those are defined below) in Γ then W-Max Ones(Γ)-2 is **APX**-hard unless W-Max Ones(Γ) is tractable. We also show some structure results for the Δ-matroid relations contained in IS_{10} and IS_{12}.

6.4.1 Definitions and Results

The previous research done on CSP(Γ)-2 (e.g., in [47, 56, 58]) has used the theory of Δ-matroids. Those objects are a generalisation of matroids and has been widely studied, cf. [15, 16]. It turns out that the complexity of W-Max Ones(Γ)-2 depends, to a large degree, on if there is a relation which is not a Δ-matroid relation in the constraint language. Δ-matroid relations are defined as follows.

Definition 6.12 (Δ-matroid relation [47])**.** *Let $R \subseteq \{0,1\}^r$ be a relation. If $\boldsymbol{x}, \boldsymbol{x'} \in \{0,1\}^r$, then $\boldsymbol{x'}$ is a* step from $\boldsymbol{x}$ to $\boldsymbol{y}$ *if $d_H(\boldsymbol{x}, \boldsymbol{x'}) = 1$ and $d_H(\boldsymbol{x}, \boldsymbol{x'}) + d_H(\boldsymbol{x'}, \boldsymbol{y}) = d_H(\boldsymbol{x}, \boldsymbol{y})$. R is a* Δ-matroid relation *if it satisfies the following two-step axiom: $\forall \boldsymbol{x}, \boldsymbol{y} \in R$ and $\forall \boldsymbol{x'}$ a step from $\boldsymbol{x}$ to $\boldsymbol{y}$, either $\boldsymbol{x'} \in R$ or $\exists \boldsymbol{x''} \in R$ which is a step from $\boldsymbol{x'}$ to $\boldsymbol{y}$.*

As an example of a Δ-matroid relation consider NAND^3. It is not hard to see that NAND^3 satisfies the two-step axiom for every pair of tuples as there is only one tuple which is absent from the relation. Other examples are the relations R_n and R_n^b for all $n, b \geq 1$ and R_n^I, for all $n \geq 1$ and all subsets I of $[n]$ with the property that if $k, k+3 \in I$, then $k+1$ or $k+2$ are contained in I. (We defined these relations in Section 6.1.) EQ^3 is the simplest example of a relation which is not a Δ-matroid relation. The main theorem of this section is the following partial classification result for W-Max Ones(Γ)-2. We say that a constraint language Γ *is* Δ*-matroid* if every relation in Γ is a Δ-matroid relation.

Theorem 6.13. *Let Γ be a conservative constraint language.*

1. *If $\Gamma \subseteq IV_2$ or $\Gamma \subseteq ID_1$, then* W-Max Ones(Γ)-2 *is in* **PO***;*
2. *else if $\Gamma \subseteq IL_2$ and,*
   - *Γ is not Δ-matroid, then* W-Max Ones(Γ)-2 *is* **APX***-complete;*
   - *otherwise,* W-Max Ones(Γ)-2 *is in* **PO***;*
3. *else if $\Gamma \subseteq ID_2$ and,*
   - *Γ is not Δ-matroid, then* W-Max Ones(Γ)-2 *is* **poly-APX***-complete;*
   - *otherwise,* W-Max Ones(Γ)-2 *is in* **PO***;*
4. *else if $\Gamma \subseteq IE_2$ and Γ is not Δ-matroid, then* W-Max Ones(Γ)-2 *is* **APX***-hard;*

5. *else if* Γ *is not* Δ*-matroid, then it is* **NP***-hard to find feasible solutions to* W-Max Ones(Γ)-2.

Part 1 of the theorem follows from the known results for W-Max Ones [9]. Part 4 follows from results for CSP(Γ)-2 [56, Theorem 4]. The other parts follow from the results in Sections 6.4.3, 6.4.4, and 6.4.5 below.

The structure results for the Δ-matroid relations in IS_{10} and IS_{12} are contained in Section 6.4.5.

6.4.2 Tractability Results for W-Max Ones(Γ)-2

Edmonds and Johnson [54] (see [134, Chapter 36] for another presentation of this method) have shown that the following integer linear programming problem is solvable in polynomial time: maximise $\boldsymbol{w}^T\boldsymbol{x}$ subject to the constraints $\boldsymbol{c} \leq \boldsymbol{x} \leq \boldsymbol{d}$, $\boldsymbol{b_1} \leq A\boldsymbol{x} \leq \boldsymbol{b_2}$ and $\boldsymbol{x}$ is an integer vector. Here A is a matrix with integer entries such that the sum of the absolute values of each column is at most 2 and $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{b_1}$, $\boldsymbol{b_2}$, and $\boldsymbol{w}$ are arbitrary real vectors of appropriate dimensions. We will denote this problem by ILP-2. With the polynomial solvability of ILP-2 it is possible to prove the tractability of a number of W-Max Ones(Γ)-2 problems. Unless explicitly stated otherwise we use ILP-2 instances such that $\boldsymbol{c} = \boldsymbol{0}$ and $\boldsymbol{d} = \boldsymbol{1}$, so $\boldsymbol{x}$ is constrained to be a 0-1 vector.

6.4.3 Classification of ID_2

In this section we will classify the complexity of the constraint languages Γ such that $\mathrm{Pol}(\Gamma) = \mathrm{Pol}(ID_2)$. We need the following lemma, which was proved by Feder [56, Theorem 3, fact 3], before we can give the proof of the hardness result.

Lemma 6.14 ([56])**.** *Let* Γ *be a constraint language such that* $\mathrm{Pol}(\Gamma) = \mathrm{Pol}(ID_2)$*, then every two-literal clause is* 2*-representable. That is,* $x \wedge y$*,* $x \vee y$*, IMPL and* NAND^2 *are* 2*-representable by* Γ*.*

We are now ready to prove the hardness lemma for the ID_2 case.

Lemma 6.15. *Let* Γ *be a constraint language such that* $\mathrm{Pol}(\Gamma) = \mathrm{Pol}(ID_2)$*. If there is a relation* $R \in \Gamma$ *which is not a* Δ*-matroid relation, then* W-Max Ones(Γ)-2 *is* **poly-APX***-complete.*

The main observations used to prove the lemma is that since $\mathrm{Pol}(\Gamma) = \mathrm{Pol}(ID_2)$, by using Lemma 6.14, we can 2-represent every two-literal clause. Furthermore, if we have access to every two-literal clause and also have a non-Δ-matroid relation then it is possible to make variables participate in three clauses, which was also proved in [56]. The hardness result then follows with a reduction from MIS.

Proof (Of Lemma 6.15). We will do an S-reduction from the **poly-APX**-complete problem MIS, which is precisely Max Ones($\{\mathrm{NAND}^2\}$).

Let $I = (V, C)$ be an arbitrary instance of MAX ONES($\{\text{NAND}^2\}$). We will construct an instance $I' = (V', C', w)$ of W-MAX ONES(Γ)-2. From Lemma 6.14 we know that we can 2-represent every two-literal clause. It is easy to modify I so that each variable occurs at most three times. For a variable $x \in V$ which occurs k times, introduce k fresh variables $y_1, y_2, \ldots, y_k$ and add the constraints $\text{IMPL}(y_1, y_2), \text{IMPL}(y_2, y_3), \ldots, \text{IMPL}(y_k, y_1)$. Each occurrence of x is then replaced with one of the y_i variables. In every solution each of the y_i variables will obtain the same value, furthermore they occur three times each. Hence, if we can create a construction which allows us to let a variable participate in three clauses we are done with our reduction.

In [56, Theorem 4] Feder showed that given a relation which is not a Δ-matroid we can make variables participate in three clauses if we have access to all two-literal clauses. We adapt Feder's proof to our setting. Let $R \in \Gamma$ be a non-Δ-matroid relation. There are $\boldsymbol{x}, \boldsymbol{y} \in R$ and $k \in [n]$ such that $\boldsymbol{x} \oplus k$ is a step from $\boldsymbol{x}$ to $\boldsymbol{y}$ and for all $k' \in [n], \boldsymbol{x}(k') \neq \boldsymbol{y}(k')$ we have that $\boldsymbol{x} \oplus \{k, k'\} \notin R$.

We define a new non-Δ-matroid relation with the 2-representation

$$R(v_1, \ldots, v_n) \wedge \bigwedge_{i \in [n]: \boldsymbol{x}(i) = \boldsymbol{y}(i)} v_i = \boldsymbol{x}(i).$$

It follows that we can assume that $\boldsymbol{x}(i) \neq \boldsymbol{y}(i)$ for all $i \in [n]$.

Define K to be a subset of $[n]$ of minimal cardinality such that $k \in K$ and the tuple $\boldsymbol{x} \oplus K$ is contained in R. Note that we must have $|K| \geq 3$. By restricting each coordinate $i \in [n] \setminus K$ to $\boldsymbol{x}(i)$ we can assume that K are all the coordinates. We can now get a non-Δ-matroid relation of arity three by the 2-representation

$$\exists v_4, \ldots, v_n : R(v_1, v_2, \ldots, v_n)$$

(assuming that $K = [n]$). From now on we assume that R is a non-Δ-matroid relation of arity three.

As we can 2-represent every two-literal clause we can assume that $\boldsymbol{x} = (0, 0, 0)$ and $\boldsymbol{y} = (1, 1, 1)$. (To see this, let us look at the example $R = \{(1, 0, 0), (0, 1, 1)\}$. If some variable v participates in the clauses $\neg v \vee \neg x$, $\neg v \vee y$ and $v \vee \neg z$, then we can replace these clauses by $R(v_1, v_2, v_3)$, $v_1 \vee \neg x$, $\neg v_2 \vee y$, $v_3 \vee \neg z$, where v_1, v_2, and v_3 are fresh variables and set $w(v_2) = 1$ and $w(v_1) = w(v_3) = 0$. Note that we cannot negate all variables, but this is not needed as $(0, 0, 0) \in R$ if and only if $(1, 1, 1) \in R$.)

Assume, without loss of generality, that $k = 1$. As R is a non-Δ-matroid relation it follows that $(1, 0, 0)$, $(1, 1, 0)$, $(1, 0, 1) \notin R$, furthermore $\boldsymbol{a} = (0, 1, 0)$, $\boldsymbol{b} = (0, 0, 1)$, and $\boldsymbol{c} = (0, 1, 1)$ are possibly in R.

If none of $\boldsymbol{a}, \boldsymbol{b}$, and $\boldsymbol{c}$ are in R, then $R = \{(0, 0, 0), (1, 1, 1)\}$ and we can make variables participate in three clauses as in the example above.

If one of $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ are in R, then $R(x, y, z) \iff x \leq y = z$ or $R(x, y, z) \iff x = y \leq z$ (if the variables are reordered appropriately).

In this case we can use a circle of constraints of the form $R(x_1, y_1, z_1)$, $R(z_1, y_2, z_2)$, $R(z_2, y_3, x_1)$ to 2-represent EQ^3 where $w(x_1)$, $w(z_1)$, and $w(z_2)$ are 0.

If $\boldsymbol{a}$ and $\boldsymbol{c}$ are contained in R or $\boldsymbol{b}$ and $\boldsymbol{c}$ are contained in R, then $R(x, y, z) \iff x \leq y \leq z$ (up to reordering) and we can use the same construction again.

There are now two cases left, when only $\boldsymbol{a}$ and $\boldsymbol{b}$ are contained in R and when $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ are contained in R. In the second case $R(x, y, z) \iff x \leq y, z$ and in the first case we have $R(x, y, z) \iff x = \min(y, z)$. The latter can be reduced to the former by $x = \min(y', z')$, $y' \leq y$, $z' \leq z$ (note that $y' \leq y$ and $z' \leq z$ are two-literal clauses).

So $R(x, y, z) \iff x \leq y, z$. We can assume, without loss of generality, that each variable v occurs at least once in its negated form (if some variable occurs only in its positive form, then we can set this variable to 1 and remove all clauses where it appears). We now replace the three clauses where v appears according to the following table. (In the table x, y, and z are literals.)

Case	Clauses			Replacement		
1	$\neg v \vee x$,	$v \vee y$,	$v \vee z$	$v_1 \vee x$, $v_1 \leq v_2, v_3$,	$\neg v_2 \vee y$, $\neg v_1 \vee \neg v_4$	$\neg v_3 \vee z$,
2	$v \vee x$,	$\neg v \vee y$,	$\neg v \vee z$	$v_1 \vee x$, $v_1 \leq v_2, v_3$	$\neg v_2 \vee y$,	$\neg v_3 \vee z$,
3	$\neg v \vee x$,	$\neg v \vee y$,	$\neg v \vee z$	$\neg v_1 \vee x$, $v_1 \leq v_2, v_3$	$\neg v_2 \vee y$,	$\neg v_3 \vee z$,

In the first case, when v_4 is introduced, we set $w(v_4) = 1$ and $w(v_1) = w(v_2) = w(v_3) = 0$. In the other two cases we set $w(v_1) = 1$ and $w(v_2) = w(v_3) = 0$.

First note that if v, x, y, z is a solution to the clauses with measure m, then there is an assignment to v_1, v_2, v_3 (and v_4 in Case 1), x, y, and z such that this assignment is a solution to the replacement with the same measure. In Case 1 we let $v_1 = v_2 = v_3 = \neg v$ and $v_4 = v$. It is not hard to see that if v, x, y, z is a solution to the clauses, then this assignment is a solution to the replacement with the same measure. In Case 2 and Case 3 we let $v_1 = v_2 = v_3 = v$. It is clear that if v, x, y, z is a solution to the clauses, then this assignment is a solution to the replacement with the same measure.

Now, let $v_1, v_2, v_3, v_4, x, y, z$ be a solution to the replacement in Case 1. If $v_1 = 1$, then $v_2 = v_3 = 1$, $v_4 = 0$ and $v = 0, x, y, z$ is a solution to the clauses with the same measure. If $v_1 = 0$, then $x = 1$ and $v = 1, x, y, z$ is a solution to the clauses with a measure not less than the measure of the solution to the replacement.

Let v_1, v_2, v_3, x, y, z be a solution to the replacement in Case 2. If we let $v = v_1$, then v, x, y, z is a solution to the clauses. Furthermore, the measures of these two solutions are the same. Case 3 works in the same way.

We have shown that from any solution s' to I' one can construct a solution s to I such that $m(I,s) \geq m(I',s')$. Furthermore, for any solution s to I there is a solution s' to I' such that $m(I,s) = m(I',s')$. In particular $\text{OPT}(I) = \text{OPT}(I')$. So if s' is an r-approximate solution, then so is s. Hence, the construction above is an S-reduction from MIS to W-MAX ONES(Γ)-2. □

We will now define a constraint language denoted by $\mathcal{Q}$. We will later prove that W-MAX ONES($\mathcal{Q}$)-2 is in **PO**. $\mathcal{Q}$ is the smallest constraint language such that:

- $\emptyset$, c_0, c_1, EQ^2 and $\{(0,1),(1,0)\}$ are in $\mathcal{Q}$;
- for any positive integer n the relation $\{\boldsymbol{t} \in \{0,1\}^n \mid |\boldsymbol{t}| \leq 1\}$ is contained in $\mathcal{Q}$;
- if $R, R' \in \mathcal{Q}$ then their Cartesian product $\{(\boldsymbol{t},\boldsymbol{t'}) \mid \boldsymbol{t} \in R, \boldsymbol{t'} \in R'\}$ is also in $\mathcal{Q}$;
- if $R \in \mathcal{Q}$ and n is the arity of R then $R \oplus A \in \mathcal{Q}$ for every $A \subseteq [n]$;
- if $R \in \mathcal{Q}$, n is the arity of R and $f : [n] \to [n]$ is a permutation on $[n]$ then $\{(\boldsymbol{t}(f(1)), \boldsymbol{t}(f(2)), \ldots, \boldsymbol{t}(f(n))) \mid \boldsymbol{t} \in R\}$ is in $\mathcal{Q}$.

The following lemma describes the structure of the Δ-matroid relations in ID_2. The proof is fairly long and a bit tedious but the ideas underlying the proof are quite simple.

Lemma 6.16. *If $R \in ID_2$ is a Δ-matroid relation, then $R \in \mathcal{Q}$.*

Proof. For a relation $R \in \mathcal{Q}$ if R can be decomposed (possibly after a permutation of the coordinates of R) into a Cartesian product of other relations, $P_1, P_2, \ldots, P_n \in \mathcal{Q}$ then $P_1, P_2, \ldots, P_n$ will be called the *factors* of R.

In this proof we will denote the majority function by m, i.e., $m(x,y,z) = (x \wedge y) \vee (y \wedge z) \vee (x \wedge z)$. Note that every relation in ID_2 is invariant under m. Let R be a relation which contradicts the lemma, i.e., $R \in ID_2$, R is a Δ-matroid and $R \notin \mathcal{Q}$. Let n be the arity of R. We can assume without loss of generality that R consists of one factor, i.e., it is not possible to decompose R into a Cartesian product of other relations. In particular, for every $i \in [n]$ there are $\boldsymbol{t}, \boldsymbol{t'} \in R$ such that $\boldsymbol{t}(i) = 0$ and $\boldsymbol{t'}(i) = 1$.

As every relation of arity less than or equal to two is in $\mathcal{Q}$ we can assume that $n \geq 3$. If for every pair $\boldsymbol{t}, \boldsymbol{t'} \in R$ and every step $\boldsymbol{s}$ from $\boldsymbol{t}$ to $\boldsymbol{t'}$ we have $\boldsymbol{s} \in R$, then as no coordinate is constant, we must have $(0,0,\ldots,0) \in R$ and $(1,1,\ldots,1) \in R$. However, if $(0,0,\ldots,0), (1,1,\ldots,1) \in R$ and every step from the former to the latter is in R then every tuple with one coordinate set to 1 is in R, too. We can continue this way and get every tuple with two coordinates set to one and then every tuple with k coordinates set to 1 for $k \in [n]$. Hence, we must have $R = \{0,1\}^n \in \mathcal{Q}$.

We can therefore assume that there is a coordinate l such that the step $\boldsymbol{s} = \boldsymbol{t} \oplus l$ from $\boldsymbol{t}$ to $\boldsymbol{t'}$ is not in R. Then, as R is a Δ-matroid relation, there exists another coordinate K such that $\boldsymbol{s} \oplus \{K\} \in R$ is a step from $\boldsymbol{s}$ to $\boldsymbol{t'}$. Let X denote the set of coordinates i such that $\boldsymbol{t} \oplus i \notin R$ but $\boldsymbol{t} \oplus \{K, i\} \in R$, furthermore choose $\boldsymbol{t}$ and K such that $|X|$ is maximised and let $X' = X \cup \{K\}$.

Our goal in the rest of the proof is to show that if $X' = [n]$ then $R \in \mathcal{Q}$ and otherwise it is possible to decompose R into a Cartesian product with $\mathrm{pr}_{X'} R$ in one factor and $\mathrm{pr}_{[n]\setminus X'} R$ in the other factor. As we have assumed that R cannot be decomposed into a Cartesian product we get a contradiction and hence the relation R cannot exist.

Case 1: $|X'| = 2$.

We will start with the case when $|X'| = 2$. Assume, without loss of generality, that $X' = \{x, K\}$ then $\boldsymbol{t}, \boldsymbol{t} \oplus \{x, K\} \in R$ and $\boldsymbol{t} \oplus x \notin R$. We will now prove that we cannot have any tuples $\boldsymbol{v}$ in R such that $\mathrm{pr}_{X'} \boldsymbol{v} = \mathrm{pr}_{X'} (\boldsymbol{t} \oplus x)$.

Claim 1.A. If $\boldsymbol{v} \in R$, then $\mathrm{pr}_{X'} \boldsymbol{v} \neq \mathrm{pr}_{X'} (\boldsymbol{t} \oplus x)$.

Proof. Assume, for the sake of contradiction, that $\boldsymbol{v} \in R$ and $\mathrm{pr}_{X'} \boldsymbol{v} = \mathrm{pr}_{X'} (\boldsymbol{t} \oplus x)$. Then, $m(\boldsymbol{v}, \boldsymbol{t}, \boldsymbol{t} \oplus \{x, K\}) = \boldsymbol{w} \in R$ due to the fact that $R \in ID_2$ and m is a polymorphism of ID_2. Furthermore, $\boldsymbol{w}$ must have the same value as $\boldsymbol{t}$ on every coordinate except for possibly x and K, this follows from the fact that $\boldsymbol{t}$ has the same value as $\boldsymbol{t} \oplus \{x, K\}$ on every coordinate except for x and K. Hence, the only coordinates for which we do not know the value of $\boldsymbol{w}$ are x and K. However, $\boldsymbol{v}(K) = \boldsymbol{t}(K)$ (due do the construction of $\boldsymbol{v}$ and the fact that $K \in X'$). Hence we must get $\boldsymbol{w}(K) = \boldsymbol{t}(K)$. For $\boldsymbol{w}(x)$ note that $\boldsymbol{v}(x) = (\boldsymbol{t} \oplus \{x, K\})(x)$, hence $\boldsymbol{w}(x) = (\boldsymbol{t} \oplus x)(x)$. We can finally conclude $\boldsymbol{w} = \boldsymbol{t} \oplus x$ which is a contradiction with the construction of X'.

Similar arguments as the above will be used repeatedly in this proof. However, the presentation will not be as detailed as the one above.

We split the remaining part of Case 1 into two subcases, when $\boldsymbol{t} \oplus K \notin R$ (Subcase 1.1) and $\boldsymbol{t} \oplus K \in R$ (Subcase 1.2).

Subcase 1.1: $\boldsymbol{t} \oplus K \notin R$.

Assume that $\boldsymbol{t} \oplus K \notin R$, then $\mathrm{pr}_{X'} (\boldsymbol{t} \oplus K) \notin \mathrm{pr}_{X'} R$, because given a tuple $\boldsymbol{v}$ such that $\mathrm{pr}_{X'} \boldsymbol{v} = \mathrm{pr}_{X'} (\boldsymbol{t} \oplus K)$ then $m(\boldsymbol{t}, \boldsymbol{t} \oplus \{x, K\}, \boldsymbol{v}) = \boldsymbol{t} \oplus K$, which is not in R by the assumption we made. Furthermore, for any tuple $\boldsymbol{v} \in R$, $\boldsymbol{v} \oplus x$ is a step from $\boldsymbol{v}$ to either $\boldsymbol{t}$ or $\boldsymbol{t} \oplus \{x, K\}$, but $\boldsymbol{v} \oplus x \notin R$ (because either $\mathrm{pr}_{X'} \boldsymbol{v} = \mathrm{pr}_{X'} \boldsymbol{t}$ which implies $\boldsymbol{v} \oplus x \notin R$ by Claim 1.A or $\mathrm{pr}_{X'} \boldsymbol{v} = \mathrm{pr}_{X'} (\boldsymbol{t} \oplus \{x, K\})$ which implies $\mathrm{pr}_{X'} (\boldsymbol{v} \oplus x) = \mathrm{pr}_{X'} (\boldsymbol{t} \oplus K) \notin \mathrm{pr}_{X'} R$ by the assumption of this subcase).

The only way to get from $\boldsymbol{v} \oplus x$ to something which is in R is by flipping coordinate K, hence $\boldsymbol{v} \oplus \{x, K\} \in R$. This is the end of the case when $\boldsymbol{t} \oplus K \notin R$, because what we have proved above is that R can be decomposed

into a Cartesian product with the coordinates X' in one factor and $[n] \setminus X'$ in the other factor.

Subcase 1.2: $\boldsymbol{t} \oplus K \in R$**.**

By Claim 1.A we know that $\mathrm{pr}_{X'}\ (\boldsymbol{t} \oplus x) \notin \mathrm{pr}_{X'}\ R$. We will now show that for any $\boldsymbol{v} \in R$ such that, $\mathrm{pr}_{X'}\ \boldsymbol{v}$ is either $\mathrm{pr}_{X'}\ \boldsymbol{t}$ or $\mathrm{pr}_{X'}\ (\boldsymbol{t} \oplus \{x, K\})$, we have $\boldsymbol{v} \oplus \{x, K\} \in R$.

To this end, let $\boldsymbol{v}$ be an arbitrary tuple in R satisfying one of the conditions above. We will consider the two possible cases separately.

- If $\mathrm{pr}_{X'}\ \boldsymbol{v} = \mathrm{pr}_{X'}\ \boldsymbol{t}$ then $\boldsymbol{v} \oplus x$ is a step from $\boldsymbol{v}$ to $\boldsymbol{t} \oplus \{x, K\}$ and $\boldsymbol{v} \oplus x \notin R$. Furthermore, from Claim 1.A it follows that the only way to get into R is by flipping K hence $\boldsymbol{v} \oplus \{x, K\} \in R$.
- If $\mathrm{pr}_{X'}\ \boldsymbol{v} = \mathrm{pr}_{X'}\ (\boldsymbol{t} \oplus \{x, K\})$ then $\boldsymbol{v} \oplus K$ is a step from $\boldsymbol{v}$ to $\boldsymbol{t}$ and $\boldsymbol{v} \oplus K \notin R$ (note that $\mathrm{pr}_{X'}\ (\boldsymbol{v} \oplus K) = \mathrm{pr}_{X'}\ (\boldsymbol{t} \oplus x)$, so the claim follows from Claim 1.A). Furthermore, the only way to get into R is by flipping x hence $\boldsymbol{v} \oplus \{x, K\} \in R$.

Now, let $\boldsymbol{v}$ be an arbitrary tuple in R such that $\mathrm{pr}_{X'}\ \boldsymbol{v} = \mathrm{pr}_{X'}\ (\boldsymbol{t} \oplus K)$ then $\boldsymbol{v} \oplus K$ is a step from $\boldsymbol{v}$ to $\boldsymbol{t}$. If $\boldsymbol{v} \oplus K \in R$ or $\boldsymbol{v} \oplus x \in R$ then we are done with this step, so assume that $\boldsymbol{v} \oplus K, \boldsymbol{v} \oplus x \notin R$. However, as R is a Δ-matroid relation there has to exist a coordinate $l \neq K, x$ such that $\boldsymbol{v} \oplus \{K, l\} \in R$ (we cannot have $l = x$ due to Claim 1.A). Then we get, $\mathrm{pr}_{X'}\ (\boldsymbol{v} \oplus \{K, l\}) = \mathrm{pr}_{X'}\ \boldsymbol{t}$ which implies $\boldsymbol{v} \oplus \{x, l\} \in R$ by the argument above. However, this means that $|X|$ is not maximal as we could have chosen $\boldsymbol{v}$, l and X' instead of $\boldsymbol{t}$, K and X, respectively. We conclude that $\boldsymbol{v} \oplus K \in R$ or $\boldsymbol{v} \oplus x \in R$, which implies $\boldsymbol{v} \oplus K \in R$ and $\boldsymbol{v} \oplus x \in R$.

Finally, let $\boldsymbol{v}$ be an arbitrary tuple in R such that $\mathrm{pr}_{X'}\ \boldsymbol{v} = \mathrm{pr}_{X'}\ \boldsymbol{t}$ then $\boldsymbol{v} \oplus K \in R$. To see this note that $m(\boldsymbol{t} \oplus K, \boldsymbol{v}, \boldsymbol{v} \oplus \{x, K\}) = \boldsymbol{v} \oplus K$.

We have now proved that R can be decomposed into a Cartesian product with the coordinates X' in one factor and $[n] \setminus X'$ in the other factor for this case too. As we have assumed that the arity of R is strictly greater than two we have $X' \neq [n]$. Hence, $[n] \setminus X' \neq \emptyset$.

Case 2: $|X'| > 2$.

We will now establish a number of properties which R must enjoy. Assuming that $X' \neq [n]$, our main goal is still to show that R can be decomposed into a Cartesian product with X' in one factor and $[n] \setminus X'$ in one factor. If $X' = [n]$ we will show that $R \in \mathcal{Q}$.

Claim 2.A. If $d_H(\mathrm{pr}_X\ \boldsymbol{x}, \mathrm{pr}_X\ \boldsymbol{t}) \geq 1$ **and** $\boldsymbol{x}(K) = \boldsymbol{t}(K)$**, then** $\boldsymbol{x} \notin R$**.**

Proof. Let $\boldsymbol{x}$ be a tuple which satisfies the condition in the claim, assume that $\boldsymbol{x} \in R$, and let $i \in X$ be a coordinate where $\boldsymbol{x}$ differs from $\boldsymbol{t}$. By the construction of X we have that $\boldsymbol{t} \oplus \{K, i\} \in R$, hence we get $m(\boldsymbol{t}, \boldsymbol{t} \oplus \{K, i\}, \boldsymbol{x}) = \boldsymbol{t} \oplus \{i\} \in R$, which is a contradiction.

Claim 2.B. If $d_H(\mathrm{pr}_X\ \boldsymbol{x}, \mathrm{pr}_X\ \boldsymbol{t}) \geq 2$, then $\boldsymbol{x} \notin R$.
Proof. We will prove this claim by induction on $d = d_H(\mathrm{pr}_X\ \boldsymbol{x}, \mathrm{pr}_X\ \boldsymbol{t})$. For the base case, let $d = 2$. Let $x \in X$ be some coordinate such that $\boldsymbol{x}(x) \neq \boldsymbol{t}(x)$, if $\boldsymbol{x} \in R$ and $\boldsymbol{x}(K) = \boldsymbol{t}(K)$ then $m(\boldsymbol{t}, \boldsymbol{x}, \boldsymbol{t} \oplus \{x, K\}) = \boldsymbol{t} \oplus x \notin R$. Hence $\boldsymbol{x}(K) = \boldsymbol{t}(K)$ is not possible.

On the other hand if $\boldsymbol{x}(K) \neq \boldsymbol{t}(K)$ then $\boldsymbol{x} \oplus K$ is a step from $\boldsymbol{x}$ to $\boldsymbol{t}$. By the argument in the preceding paragraph we get $\boldsymbol{x} \oplus K \notin R$ (note that $K \notin X$ hence we have $d_H(\mathrm{pr}_X\ (\boldsymbol{x} \oplus K), \mathrm{pr}_X\ \boldsymbol{t}) = d$). Furthermore as R is a Δ-matroid relation we can flip some coordinate $l \in X$ such that $\boldsymbol{t}(l) \neq \boldsymbol{x}(l)$ to get a tuple which is in R ($l \notin X$ will not work as the argument in the preceding paragraph still applies in that case). However, $d_H(\mathrm{pr}_X\ (\boldsymbol{x} \oplus \{K, l\}), \mathrm{pr}_X\ \boldsymbol{t}) = d - 1 = 1$ hence by Claim 2.A we get a contradiction.

Now, assume that Claim 2.B holds for $d = d'$. We will prove that it also holds for $d = d' + 1$ such that $2 < d \leq |X|$. Note that we can use exactly the same argument as above except for the very last sentence in which we appeal to Claim 2.B with $d = d'$ instead of using Claim 2.A. As we have assumed that Claim 2.B holds for $d = d'$ we are done.

Claim 2.C. There is a tuple $\boldsymbol{z} \in \mathrm{pr}_{X'}\ R$ such that for any tuple $\boldsymbol{x} \in R$ we have $d_H(\mathrm{pr}_{X'}\ \boldsymbol{x}, \boldsymbol{z}) \leq 1$.
Proof. As $|X'| > 2$ (this is the assumption we make in Case 2), there are tuples $\boldsymbol{t} \oplus \{i, K\}$ and $\boldsymbol{t} \oplus \{j, K\}$ for distinct $i, j, K \in X'$ in R. Hence, the tuple $\boldsymbol{z'} = m(\boldsymbol{t}, \boldsymbol{t} \oplus \{i, K\}, \boldsymbol{t} \oplus \{j, K\}) = \boldsymbol{t} \oplus K \in R$. Let $\boldsymbol{z} = \mathrm{pr}_{X'}\ \boldsymbol{z'}$. We will now show that $d_H(\boldsymbol{z}, \mathrm{pr}_{X'}\ \boldsymbol{x}) \leq 1$ for every tuple $\boldsymbol{x}$ in R. To this end, let $\boldsymbol{x}$ be an arbitrary tuple in R. By Claim 2.B we must have $d_H(\mathrm{pr}_X\ \boldsymbol{x}, \mathrm{pr}_X\ \boldsymbol{t}) \leq 1$, furthermore if $\boldsymbol{x}(K) = \boldsymbol{z}(K) \neq \boldsymbol{t}(K)$ then we are done as $d_H(\mathrm{pr}_{X'}\ \boldsymbol{x}, \boldsymbol{z}) \leq 1$ in this case. On the other hand, if $\boldsymbol{x}(K) = \boldsymbol{t}(K)$ then Claim 2.A tells us that we must have $d_H(\mathrm{pr}_X\ \boldsymbol{x}, \mathrm{pr}_X\ \boldsymbol{t}) = 0$ in which case Claim 2.C follows.

Claim 2.D. If $\boldsymbol{x} \in R$ and $\mathrm{pr}_{X'}\ \boldsymbol{x} = \boldsymbol{z} \oplus i$ for some $i \in X'$, then $\boldsymbol{x} \oplus \{i, j\} \in R$ for every $j \in X', j \neq i$.
Proof. Given $j \in X', j \neq i$, there is at least one tuple $\boldsymbol{v} \in R$ such that $\boldsymbol{v}(j) \neq \boldsymbol{x}(j)$ since otherwise the coordinate j would be constant and R could be decomposed into a Cartesian product. Hence, $\boldsymbol{x'} = \boldsymbol{x} \oplus j$ is a step from $\boldsymbol{x}$ to $\boldsymbol{v}$, but Claim 2.C tells us that $\boldsymbol{x'} \notin R$ and the only way to fulfil the two-step axiom is if $\boldsymbol{x} \oplus \{i, j\} \in R$.

Claim 2.E. If $\boldsymbol{x} \in R$ and $\mathrm{pr}_{X'}\ \boldsymbol{x} = \boldsymbol{z}$, then $\boldsymbol{x} \oplus i \in R$ for every $i \in X'$.
Proof. By Claim 2.D it is sufficient to show that $\boldsymbol{x} \oplus i \in R$ for *some* $i \in X'$. Assume for the sake of contradiction that there is a tuple $\boldsymbol{x} \in R$ such that $\mathrm{pr}_{X'}\ \boldsymbol{x} = \boldsymbol{z}$ and $\boldsymbol{x} \oplus i \notin R$ for every $i \in X'$. As there is no

coordinate which is constant in R for each $i \in X'$ there is some tuple $\boldsymbol{t_i} \in R$ such that $\boldsymbol{x} \oplus i$ is a step from $\boldsymbol{x}$ to $\boldsymbol{t_i}$. As R is a Δ-matroid relation there is, for each i, a coordinate j_i such that $\boldsymbol{t_i} \oplus \{i, j_i\} \in R$. If two of the j_i:s are different, that is, if there are p and q such that $j_p \neq j_q$, then

$$m(\boldsymbol{x}, \boldsymbol{x} \oplus \{i, j_p\}, \boldsymbol{x} \oplus \{i, j_q\}) = \boldsymbol{x} \oplus i$$

and the claim follows from Claim 2.D. On the other hand, if all j_i:s are equal then X is not of maximal cardinality as we could have chosen $\boldsymbol{x}$, j_i and X' instead of $\boldsymbol{t}$, K, and X, respectively.

Claim 2.F. If $\boldsymbol{x} \in R$ and $\mathrm{pr}_{X'}\, \boldsymbol{x} = \boldsymbol{z} \oplus i$ for some $i \in X'$, then $\boldsymbol{x} \oplus i \in R$.

Proof. Let $j \in X, j \neq i$. By Claim 2.D we have $\boldsymbol{x} \oplus \{i, j\} \in R$. We now get $m(\boldsymbol{x}, \boldsymbol{x} \oplus \{i, j\}, \boldsymbol{z}) = \boldsymbol{x} \oplus i$.

We will now prove that R can be decomposed into a Cartesian product where the coordinates X' make up one factor and $[n] \setminus X'$ make up the other factor. Our goal is to show that for any $\boldsymbol{u} \in \mathrm{pr}_{X'} R$ and $\boldsymbol{u'} \in \mathrm{pr}_{[n]\setminus X'} R$ we have $(\boldsymbol{u}, \boldsymbol{u'}) \in R$ (we have assumed that $X' = \{1, 2, \ldots, |X'|\}$ here).

To this end, let $\boldsymbol{x}$ be an arbitrary tuple in R. By Claim 2.C we have $d_H(\mathrm{pr}_{X'}\, \boldsymbol{x}, \boldsymbol{z}) \leq 1$. If $\mathrm{pr}_{X'}\, \boldsymbol{x} = \boldsymbol{z}$, then by Claim 2.E $\boldsymbol{x} \oplus i \in R$ for every $i \in X'$. In the other case, when $\mathrm{pr}_{X'}\, \boldsymbol{x} = \boldsymbol{z} \oplus j$ for some $j \in X'$ it follows from Claim 2.F that $\boldsymbol{x} \oplus j \in R$ and from Claim 2.D that $\boldsymbol{x} \oplus \{j, i\} \in R$ for every $i \in X'$.

We conclude that if $X' \neq [n]$, then R can be decomposed into a Cartesian product. On the other hand, if $X' = [n]$, then it follows from Claim 2.C that $R \in \mathcal{Q}$. □

As for the tractability part we have the following lemma.

Lemma 6.17. *Let Γ be a constraint language such that $\Gamma \subseteq ID_2$, if all relations in Γ are Δ-matroid relations then* W-MAX ONES(Γ)-2 *is in* **PO**.

The idea behind the proof is that W-MAX ONES($\mathcal{Q}$)-2 can be seen as an ILP-2 problem and is therefore solvable in polynomial time.

Proof (Of Lemma 6.17). Let I be an arbitrary instance of W-MAX ONES(Γ)-2. We will show that the problem is in **PO** by reducing it to an instance I' of ILP-2. For any relation $R \in \Gamma$ of arity n we know, from Lemma 6.16, that $R \in \mathcal{Q}$. We can assume that R is not the Cartesian product of any other two relations, because if it is then every use of R can be substituted by the factors in the Cartesian product. If R is unary we can replace $R(x)$ by $x = 0$ or $x = 1$. If $R = EQ^2$ then we can replace $R(x, y)$ by $x = y$ and if $R(x, y) \iff x \neq y$ then we replace $R(x, y)$ by $x = y - 1$.

Now, assume that none of the cases above occur. We will show that

$$R(t_1, t_2, \ldots, t_n) \iff \sum_{i=1}^{n} a_i t_i \leq b \tag{6.1}$$

for some $a_i \in \{-1, 1\}$ and integer b. Let N be set of negated coordinates of R, i.e., let $N \subseteq [n]$ such that

$$R = \{(f(t_1, 1), f(t_2, 2), \ldots, f(t_n, n)) \mid \boldsymbol{t} = (t_1, \ldots, t_n) \in \{0, 1\}^n, |\boldsymbol{t}| \leq 1\}$$

where $f : \{0, 1\} \times [n] \to \{0, 1\}$ and $f(x, i) = 1 - x$ if $i \in N$ and $f(x, i) = x$ otherwise. According to the definition of $\mathcal{Q}$, R can be written on this form. Let $a_i = -1$ if $i \in N$ and $a_i = 1$ otherwise. Furthermore, let $b = 1 - |N|$. It is now easy to verify that (6.1) holds.

As every variable occurs at most twice in I every variable occur at most twice in I' too. Furthermore, the coefficient in front of any variable in I' is either -1, 0 or 1, so the sum of the absolute values in any column in I' is bounded by 2. I' is therefore an instance of ILP-2. If we let the weight function of I' be the same as the weight function in I it easily seen that any solution s to I' is also a solution to I with the same measure. □

6.4.4 Classification of IL_2

In this section we will prove a classification result for IL_2. In this case non-Δ-matroid relations give rise to **APX**-complete problems and absence of such relations makes the problem tractable. Also for this class the tractability follows from a reduction to ILP-2.

A linear system of equations over $\mathbb{Z}_2$ (the two element field) with n equations and m variables can be represented by a matrix A, a constant column vector $\boldsymbol{b}$ and a column vector $\boldsymbol{x} = (x_1, \ldots, x_m)$ of variables. The system of equations is then given by $A\boldsymbol{x} = \boldsymbol{b}$. Assuming that the rows of A are linearly independent the set of solutions to $A\boldsymbol{x} = \boldsymbol{b}$ are

$$\left\{(\boldsymbol{x'}, \boldsymbol{x''}) \mid \boldsymbol{x'} \in \mathbb{Z}_2^m, \boldsymbol{x''} \in \mathbb{Z}_2^{n-m} \text{ and } \boldsymbol{x'}^T = A'\boldsymbol{x''}^T + \boldsymbol{b'}\right\}$$

where $\boldsymbol{x'} = (x_1, \ldots, x_m)$, $\boldsymbol{x''} = (x_{m+1}, \ldots, x_n)$ and A' and $\boldsymbol{b'}$ are suitably chosen. (If some of the rows are linearly dependent, then we can remove dependent rows until we get a system of equations with linearly independent rows. The new system will have the same set of solutions as the original one.) If there is a column in A' with more than one entry which is equal to 1 (or, equivalently more than one non-zero entry), then we say that the system of equations is *coupled*.

Lemma 6.18. *Let Γ be a conservative constraint language such that $\Gamma \subseteq IL_2$. If there is a relation $R \in \Gamma$ such that R is the set of solutions to a coupled linear system of equations over $\mathbb{Z}_2$ then* W-Max Ones(Γ) $\leq_S$ W-Max Ones(Γ)-2*, otherwise* W-Max Ones(Γ)-2 *is in* **PO**.

Proof. First note that every relation $R \in IL_2$ is the set of solutions to a linear system of equations over $\mathbb{Z}_2$ (see Table 6.1). We will start with the hardness proof. To this end we will construct a 2-representation of EQ^3. Let $R \in \Gamma$ be defined by

$$R = \left\{(\boldsymbol{x'}, \boldsymbol{x''}) \mid \boldsymbol{x'} \in \mathbb{Z}_2^m, \boldsymbol{x''} \in \mathbb{Z}_2^{n-m} \text{ and } \boldsymbol{x'}^T = A'\boldsymbol{x''}^T + \boldsymbol{b'}\right\}$$

for some n, m, A' and $\boldsymbol{b'}$. Here $\boldsymbol{x'} = (x_1, \ldots, x_m)$ and $\boldsymbol{x''} = (x_{m+1}, \ldots, x_n)$. Furthermore, we can assume that there is one column (say j) in A' with more than one entry equal to 1 (say i and i'). Hence, A'_{ij} and $A'_{i'j}$ are equal to 1. Our first implementation consists of R and a number of c_0 constraints,

$$Q(x_{j+m}, x_i, x_{i'}) \iff \exists x_1, \ldots : R(x_1, \ldots, x_n) \bigwedge_{\substack{k:\ m+1 \leq k \leq n \\ k \notin \{j+m, i, i'\}}} c_0(x_k)$$

where the existential quantification is taken over all variables x_l such that $l \in [n] \setminus \{j+m, i, i'\}$. It follows that

$$Q(x_{j+m}, x_i, x_{i'}) \iff x_i = x_{j+m} + c \wedge x_{i'} = x_{j+m} + d$$

for some constants $c, d \in \mathbb{Z}_2$ (recall that the j:th component of $\boldsymbol{x''}$ is x_{j+m} and the i and i':th components of $\boldsymbol{x'}$ are x_i and $x_{i'}$, respectively). This means that we get four cases, the first one is $Q = EQ^3$ (if $c = d = 0$), in which case we are done. By reordering of the indices the other three cases are equivalent to $Q = \{(0,0,1), (1,1,0)\}$ (if $(c,d) \in \{(1,0), (0,1), (1,1)\}$). We give a 2-representation of EQ^3 for this case. Note that

$$EQ^3(y_1, y_2, y_3) \iff \exists z, z' : Q(y_1, y_2, z) \wedge Q(y_3, z', z).$$

For the containment proof note that every relation in Γ is the set of solutions to some non-coupled linear system of equations over $\mathbb{Z}_2$. The set of feasible solutions to an instance of W-MAX ONES(Γ)-2 is therefore the set of solutions to a linear system of equations over $\mathbb{Z}_2$ with the property that every variable occurs at most twice. This problem can be solved by a reduction to ILP-2: each equation over $\mathbb{Z}_2$, $\sum_i a_i x_i = b \pmod 2$, is replaced by an equation of the form $\sum_i a_i x_i = b + 2z$ where z is a fresh variable which is unrestricted (i.e., we do not introduce any bounds for z). □

Corollary 6.19. *Let Γ be a conservative constraint language such that $\mathrm{Pol}(\Gamma) = \mathrm{Pol}(IL_2)$. If there is a relation $R \in \Gamma$ such that R is not a Δ-matroid relation then* W-MAX ONES(Γ)-2 *is* **APX***-complete, otherwise* W-MAX ONES(Γ)-2 *is in* **PO***.*

Proof. Given a constraint language Γ such that $\mathrm{Pol}(\Gamma) = \mathrm{Pol}(IL_2)$ then W-MAX ONES(Γ) is **APX**-complete [102].

It is not hard to see that for a relation $R \in \Gamma$, R is not a Δ-matroid relation if and only if R is the set of solutions to a coupled system of equations. (The "if"-part follows directly from the representation of relation Q in Lemma 6.18.) It the system is not coupled, then the relation is a Cartesian product of relations and each factor is definable by a single equation.

Hence, if there is a relation $R \in \Gamma$ such that R is not a Δ-matroid relation then we get **APX**-completeness for W-MAX ONES(Γ)-2 from Lemma 6.18. On the other hand, if there is no non-Δ-matroid relation in Γ then no relation is the set of solutions to a coupled system of equations and hence we get tractability from Lemma 6.18. □

6.4.5 IE_2, IS_{12} and IS_{10}

In this section we will investigate the complexity of the relations contained in IE_2. We do not get a complete characterisation result for the complexity of W-MAX ONES-2 in this case. However, we prove a characterisation theorem for the Δ-matroids in IS_{12} and IS_{10}. It turns out that the Δ-matroids in IS_{12} are in fact matroids and the Δ-matroids in IS_{10} are only slightly more general than matroids. We prove this in the next subsection. After that we show that every non-Δ-matroid in IE_2 implies hardness for W-MAX ONES-2.

6.4.5.1 Δ-matroid Relations in IS_{10} and IS_{12}

Lemma 6.20. *Let $R \in IS_{10}$ be a Δ-matroid relation of arity n which cannot be decomposed into a Cartesian product. Then, either $R \in IS_{12}$ or*

$$R(x_1, x_2, \ldots, x_n) \iff \sum_{i=1}^{n-1} x_i \leq 1, \bigwedge_{i=1}^{n-1} IMPL(x_i, x_n)$$

(up to a permutation of $x_1, x_2, \ldots, x_n$).

Proof. Let $R \in IS_{10}$ be a Δ-matroid relation of arity n which cannot be decomposed into a Cartesian product. As R cannot be decomposed there is no $i \in [n]$ such that $\boldsymbol{t}(i) = 1$ (or $\boldsymbol{t}(i) = 0$) for all $\boldsymbol{t} \in R$. If $R \in IS_{12}$ then the lemma follows, so assume that $R \notin IS_{12}$. Then there exists at least one constraint application in the plain representation of R which uses IMPL (see Table 6.1). Assume, without loss of generality, that $\text{IMPL}(x_1, x_n)$ occurs in the implementation. Let I be the set of all indices $i \in [n]$ such that $\text{IMPL}(x_i, x_n)$ occurs in the implementation. We can assume that $n \notin I$.

Claim A. For every $\boldsymbol{t} \in R$ we have $|\text{pr}_I\, \boldsymbol{t}| \leq 1$.

Proof. Assume that there is a tuple $\boldsymbol{t} \in R$ such that $|\text{pr}_I\, \boldsymbol{t}| > 1$. We must have $\boldsymbol{t}(n) = 1$ and $\boldsymbol{t'} = \boldsymbol{t} \oplus n$ is a step from $\boldsymbol{t}$ to $\mathbf{0} \in R$. ($\mathbf{0}$ is contained in R as no index is constant in R and R is closed under the *and* operation.) However, as $|\text{pr}_I\, \boldsymbol{t'}| > 1 > 0$ and $\boldsymbol{t'}(n) = 0$ it follows that $\boldsymbol{t'} \notin R$. Furthermore, no step from $\boldsymbol{t'}$ to $\mathbf{0}$ is in R (as $|\text{pr}_{\boldsymbol{t'}}\, I| > 1$). This contradicts the fact that R is a Δ-matroid relation.

If $I = [n-1]$ then we are done, as we have showed that R is of the required form. We will proceed by showing that $I \neq [n-1]$ implies that R can be decomposed into a Cartesian product. More specifically, we will show that R can be decomposed into a Cartesian product with the indices $I \cup \{n\}$ in one factor and $[n-1] \setminus I$ in the other factor.

Claim B. If $\boldsymbol{t} \in R$ and $\text{pr}_{I \cup \{n\}}\, \boldsymbol{t} = \mathbf{0}$, then $\boldsymbol{t} \oplus \{i, n\} \in R$ for all $i \in I$.

Proof. As no index in R is constant, there is for every $i \in I$ a tuple $\boldsymbol{t_i} \in R$ such that $\boldsymbol{t_i}(i) = 1$ (and due to the IMPL constraints we will have $\boldsymbol{t_i}(n) = 1$ too). Then, for each $i \in I$, $\boldsymbol{t} \oplus i$ is a step from $\boldsymbol{t}$ to $\boldsymbol{t_i}$,

but $\boldsymbol{t} \oplus i \notin R$ (as $(\boldsymbol{t} \oplus i)(n) = 0$). Furthermore, the only way to get into R is by flipping n, hence we must have $\boldsymbol{t} \oplus \{i, n\} \in R$ for each i.

Claim C. In the plain representation of R there is no constraint application of the form IMPL(x_i, x_j) for any $i \in I$ and $j \neq i, n$.
Proof. To see this assume, without loss of generality, that $i < j$, then $(0,0,0), (1,1,1) \in \mathrm{pr}_{\{i,j,n\}} R$ and $(1,0,0), (1,1,0), (1,0,1) \notin \mathrm{pr}_{\{i,j,n\}} R$. As $(1,0,0)$ is a step from $(0,0,0)$ to $(1,1,1)$ and neither $(1,0,0)$, $(1,1,0)$, nor $(1,0,1)$ are contained in $\mathrm{pr}_{\{i,j,n\}} R$ it follows that $\mathrm{pr}_{\{i,j,n\}} R$ is not a Δ-matroid relation. As any projection of a Δ-matroid relation is a Δ-matroid relation the claim follows.

Claim D. If $I = \{i\}$, then either the constraint application IMPL(x_n, x_i) exists in the plain representation of R or $\boldsymbol{t}[i = 0] \in R$ for any $\boldsymbol{t} \in R$.
Proof. Note that we cannot have a constraint application of the form IMPL(x_n, x_j) for any $j \neq i, n$, because such a constraint would imply IMPL(x_i, x_j) and this is impossible by Claim C. We can also assume that there is no constraint application of the form IMPL(x_j, x_i) for any $j \neq i, n$, because such a constraint would imply IMPL(x_j, x_n) and hence $j \in I$ which is not true. So the only possible IMPL constraints are IMPL(x_i, x_n) (which we know exist) and IMPL(x_n, x_i). Assume that IMPL(x_n, x_i) does not exist. The claim now follows from the observation that the only other constraints which appear in the plain representation of R is NAND and the fact that if $\boldsymbol{x} \in$ NAND and $\boldsymbol{y} \leq \boldsymbol{x}$, then $\boldsymbol{y} \in$ NAND.

Claim E. If $|I| > 1$, then for any $\boldsymbol{t} \in R$ the tuple $\boldsymbol{t'}$ defined as $\boldsymbol{t'}(i) = 0$ for $i \in I$ and $\boldsymbol{t'}(i) = \boldsymbol{t}(i)$ for $i \in [n] \setminus I$, is contained in R.
Proof. This claim is very similar to Claim D. We cannot have a constraint application of the form IMPL(x_n, x_j) for any $j \neq n$, because such a constraint application would imply IMPL(x_i, x_j) for all $i \in I$ and this is impossible by Claim C. We can also assume that there is no constraint application of the form IMPL(x_j, x_i) for any $i \in I$ and $j \neq i, n$, because such a constraint would imply IMPL(x_j, x_n) and hence $j \in I$ and then IMPL(x_j, x_i) contradicts Claim C. As the only other constraint we can use is NAND the claim follows.

Claim F. If $\boldsymbol{t} \in R$, $\mathrm{pr}_I\ \boldsymbol{t} = \boldsymbol{0}$, and $\boldsymbol{t}(n) = 1$, then $\boldsymbol{t} \oplus n \in R$ and $\boldsymbol{t} \oplus i \in R$ for each $i \in I$.
Proof. Let $\boldsymbol{t} \in R$ be an arbitrary tuple such that $\mathrm{pr}_I\ \boldsymbol{t} = \boldsymbol{0}$ and $\boldsymbol{t}(n) = 1$. Then $\boldsymbol{t} \oplus n \in R$ unless there is a constraint application of the form IMPL(x_i, x_n) for some $i \in [n] \setminus I$ and $\boldsymbol{t}(i) = 1$, however as I is the set of all such indices no such constraint application can exist. Hence, we must have $\boldsymbol{t} \oplus n \in R$. As $\mathrm{pr}_{I \cup \{n\}}\ \boldsymbol{t} \oplus n = \boldsymbol{0}$ it follows from Claim B

that $(\boldsymbol{t} \oplus n) \oplus \{i, n\} = \boldsymbol{t} \oplus i$ is contained in R.
The proof now splits into two cases.
Case 1: $|I| > 1$**.** We have proved that for an arbitrary tuple $\boldsymbol{t} \in R$ we have $|\mathrm{pr}_I\ \boldsymbol{t}| \leq 1$ (Claim A). Furthermore, by Claim E combined with Claim F the tuple $\boldsymbol{t'}$ defined by $\boldsymbol{t'}(i) = 0$ if $i \in I \cup \{n\}$ and $\boldsymbol{t'}(i) = \boldsymbol{t}(i)$ otherwise, is contained R for every $\boldsymbol{t} \in R$. By Claim B we have $\boldsymbol{t'} \oplus \{i, n\} \in R$ and by Claim B combined with Claim E we have $\boldsymbol{t'} \oplus n \in R$.

In other words, we have proved that for every tuple $\boldsymbol{t} \in R$ there is a tuple $\boldsymbol{t'} \in R$ such that $\mathrm{pr}_{[n-1]\setminus I}\ \boldsymbol{t} = \mathrm{pr}_{[n-1]\setminus I}\ \boldsymbol{t'}$, furthermore $\mathrm{pr}_{I\cup\{n\}}\ \boldsymbol{t'}$ can be chosen arbitrarily with the restrictions that the number of ones is less than or equal to two and $\boldsymbol{t'}(i) = 1$ for some $i \in I$ implies $\boldsymbol{t'}(n) = 1$. We have also proved that if $\boldsymbol{t} \in R$ then $|\mathrm{pr}_I\ \boldsymbol{t}| \leq 1$. This implies that R can be decomposed into a Cartesian product with the indices $I \cup \{n\}$ in one factor and $[n-1] \setminus I$ in the other factor.
Case 2: $|I| = 1$**.** Let $I = \{i\}$. If the second case of Claim D holds, that is, if for any $\boldsymbol{t} \in R$ the tuple $\boldsymbol{t}[i = 0]$ is contained in R, then the argument in Case 1 works. We therefore assume that the constraint application $\mathrm{IMPL}(x_n, x_i)$ exists in a representation of R. It follows that $\mathrm{pr}_{\{i,n\}}\ R = \{(0,0),(1,1)\}$.

Let $\boldsymbol{t}$ be an arbitrary tuple in R. If $\mathrm{pr}_{\{i,n\}}\ \boldsymbol{t} = (0,0)$, then by Claim B, the tuple $\boldsymbol{t} \oplus \{i, n\}$ is contained in R. On the other hand, if $\mathrm{pr}_{\{i,n\}}\ \boldsymbol{t} = (1,1)$, then by the argument used in Claim D it follows that $\boldsymbol{t} \oplus \{i, n\} \in R$. Hence, R can be decomposed into a Cartesian product with the indices $\{i, n\}$ in one factor and $[n-1] \setminus \{i\}$ in the other factor. □

The lemma above tells us that the only Δ-matroid relations which cannot be decomposed into a Cartesian product and are contained in IS_{10} but not in IS_{12} are of the following form:

$$\begin{array}{{cccc}}\{(0 & 0 & 0 & 0),\\ (1 & 0 & 0 & 1),\\ (0 & 1 & 0 & 1),\\ (0 & 0 & 1 & 1),\\ (0 & 0 & 0 & 1)\}\end{array}$$

Such relations can be represented in ILP-2 problems by the inequality

$$x_1 + x_2 + \ldots + x_{n-1} \leq x_n.$$

A relation over a boolean domain naturally corresponds to a set system. A relation R of arity n is a set of tuples, each of length n. The set system which corresponds to R is a pair (E, I) where $|E| = n$ and I is a collection of subsets of E. More precisely, we have $E = [n]$ and $\{i \in [n] \mid t_i = 1, \boldsymbol{t} = (t_1, t_2, \ldots, t_n)\} \in I$ if and only if $\boldsymbol{t} \in R$. Hence, each tuple $\boldsymbol{t} = (t_1, t_2, \ldots, t_n)$ in the relation corresponds to a set A in I and if $t_i = 1$ then $i \in A$. It is sometimes more convenient to work with the set system instead of with the relation.

A *matroid* is a set system (E, I) which satisfies

(I) $\emptyset \in I$ and $|I| > 1$,

(II) if $A \in I$ and $B \subseteq A$ then $B \in I$, and

(III) if $A, B \in R$ and $|A| = |B| + 1$, then there is an element $x \in A \setminus B$ such that $B \cup \{x\} \in I$.

A *matroid relation* is a relation where the corresponding set system satisfies (I), (II) and (III).

For two tuples $\boldsymbol{t} = (t_1, t_2, \ldots, t_n)$ and $\boldsymbol{t'} = (t'_1, t'_2, \ldots, t'_n)$ we write $\boldsymbol{t} \leq \boldsymbol{t'}$ if $t_i \leq t'_i$ for all $i \in [n]$. We say that a relation R is *downward closed* if $\boldsymbol{t} \in R$ and $\boldsymbol{t'} \leq \boldsymbol{t}$ implies $\boldsymbol{t'} \in R$. Note that downward closedness for a relation is exactly the same thing as (II) in the definition above for the corresponding set system.

Lemma 6.21. *Let $R \in IS_{12}$ be a Δ-matroid relation of arity n such that*

- *there is no subset X of $[n]$ such that $\mathrm{pr}_X\, R = EQ^2$ and*
- *there is no $x \in [n]$ such that $\mathrm{pr}_{\{x\}}\, R = c_1$,*

then R is a matroid relation.

Proof. To simplify the presentation we will use the set system perspective of the relation, instead of looking at it as a set of tuples. Let (E, I) be the set system which corresponds to R. From the fact that $c_1, EQ^2, \{\mathrm{NAND}^k \mid k \in \mathbb{N}\}$ is a plain basis for IS_{12} and the assumptions in the lemma it follows that R is downwards closed. Hence, (I) and (II) are satisfied by R and we only need to verify (III). Let $A, B \in I$ be two sets such that $|A| = |B| + 1$. If $B \subseteq A$ then we are done, so assume that $B \not\subseteq A$.

We will show, using the Δ-matroid properties of R, that we can take a step from B towards A such that $B \cup \{j\} \in I$, for some $j \in A \setminus B$. Let $X = B \setminus A$ and let $i \in X$, then $A \triangle \{i, k\} \in I$ for some $k \in A \triangle B$. If $k \in X$ there is a subset A' of $A \triangle \{i, k\}$ such that $|A'| = |B| + 1$ and $|B \setminus A'| < |X|$ and $A' \setminus B \subseteq A \setminus B$. In this case we can repeat the argument with A' and B instead of A and B. We can therefore assume that $k \in A \setminus B$. We will now divide the proof into two cases, depending on the cardinality of X.
Case A: $|X| = 0$**.** It follows that $B \subseteq A$ which contradicts the assumption $B \not\subseteq A$.
Case B: $|X| > 0$**.** In this case $|A \setminus B| \geq |X| + 1$. Let $A' = A \triangle \{i, k\}$, we have $A' \in I$. Furthermore, $|A'| = |B| + 1$ and $|B \setminus A'| < |X|$ and $A' \setminus B \subseteq A \setminus B$, hence repeating this procedure with A', B instead of A, B will eventually get us to Case A. □

Lemma 6.22. *Let R be a matroid relation, then $R \in IS_{12}$.*

Proof. As noted before, property (II) in the definition of a matroid is the same thing as downward closedness. Remember that IS_{12} contains those relations which are closed under $f(x, y, z) = x \wedge (y \vee \neg z)$.

Given a matroid relation R and three tuples $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in R$ let $\boldsymbol{t} = f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. We have $\boldsymbol{t} \leq \boldsymbol{x}$ and as R is downward closed we must have $\boldsymbol{t} \in R$. Hence, R is closed under f which implies $R \in IS_{12}$. □

6.4.5.2 Hardness Results for IE_2

In this section we will prove that if $\Gamma \subseteq IE_2$ and there is a non-Δ-matroid relation in Γ, then W-Max Ones(Γ)-2 is **APX**-hard. The main part of our hardness result is the following lemma.

Lemma 6.23. *Let* $R(x_1, x_2, x_3) \iff NAND^2(x_1, x_2) \wedge NAND^2(x_2, x_3)$, *then* W-Max Ones($\{c_0, c_1, R\}$)-2 *is* **APX**-*complete.*

Note that R is not a Δ-matroid relation. With Lemma 6.23 and a careful enumeration of the types of non-Δ-matroid relations that exists in IE_2, we can deduce the desired result: if there is a non-Δ-matroid relation in the constraint language, then W-Max Ones($\cdot$)-2 is **APX**-hard. The proof builds upon the work in [12, 56, 99].

Proof (Of Lemma 6.23). For the containment note that the algorithm in Lemma 6.6 can be used, as an instance of W-Max Ones($\{c_0, c_1, R\}$)-2 can easily be reduced to an instance of W-Max Ones($\{c_0, c_1, \text{NAND}^2\}$)-4.

We will do an L-reduction from Max 2SAT-3 (i.e., Max 2SAT where every variable occurs at most three times), which is **APX**-complete [9, Chapter 8]. The reduction is based on Theorem 1 in [12], which in turn is based on some of Viggo Kann's work on 3-dimensional matching [99].

We will do the reduction in two steps, we will first reduce Max 2SAT-3, to a restricted variant of MIS-3. More precisely the graphs produced by the reduction will have maximum degree three and it will be possible to "cover" the graphs with R (we will come back to this soon).

Let $I = (V, C)$ be an arbitrary instance of Max 2SAT-3. We will construct an instance $I' = (G, w)$, where $G = (V', E')$ and $w : V' \to \mathbb{Q}$, of weighted maximum independent set. We can assume, without loss of generality, that each variable in I occurs at least once unnegated and at least once negated. For a node $v \in V$ construct four paths with three nodes each. Sequentially label the nodes in path number x by p_{x1}, p_{x2}, p_{x3}. Construct three complete binary trees with four leaves each and label the roots of the trees with $v_1, \neg v_1, v_2$ (or $v_1, \neg v_1, \neg v_2$ if v occurs once unnegated and twice negated). Finally, identify the leaves of each of the trees with the nodes in the paths with similar labels, where two labels p_{xy} and p_{uv} are similar if $y = v$. Figure 6.2 contains this gadget for our example variable, v.

Let the w be defined as follows, $w(p_{12}) = w(p_{22}) = w(p_{32}) = w(p_{42}) = 2.25$, $w(x_{21}) = w(x_{22}) = 2$ and $w(\cdot) = 1$ otherwise.

We use $X(v)$ to denote the nodes in these paths and trees which are associated with the variable v. Furthermore, let X be the disjoint union of the $X(v)$'s when v ranges over all variables. Let S be a solution to the independent set problem for X and let v be a variable. The solution S will

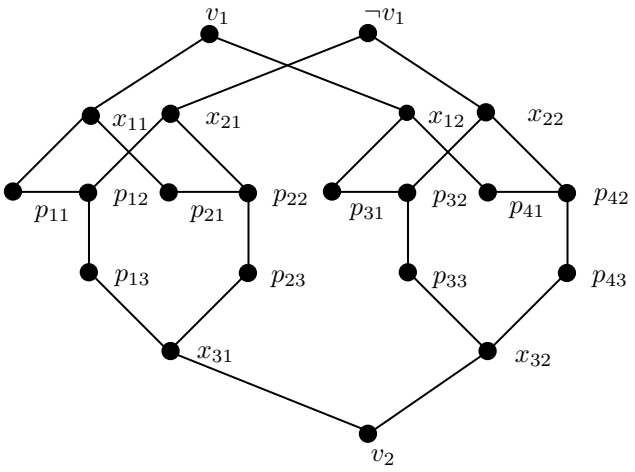


Figure 6.2: The graph gadget for the variable v which occurs three times, two times unnegated and one time negated.

be called *consistent for* v if we have $v_1, v_2 \in S$ and $\neg v_1 \notin S$ or vice versa (i.e., $v_1, v_2 \notin S$ and $\neg v_1 \in S$). We say that S is consistent if it is consistent for v for every variable v.

It is not hard to verify (e.g., with a computer assisted search) that the optimal solutions to X are consistent. Furthermore, there is an optimal solution S such that $v_1, v_2 \in S$ and $\neg v_1 \notin S$ and an optimal solution S' such that $v_1, v_2 \notin S'$ and $\neg v_1 \in S'$.

For each clause $c \in C$, containing the literals l_1 and l_2, add two fresh nodes c and c' to G'. Connect c and c' with an edge and connect c with the node which is labelled with $\neg l_1$ (one of the roots of the trees). Do the same thing for c' and $\neg l_2$. Let $w(c) = w(c') = 1$.

We now show that given a solution to I' it is possible to construct a consistent solution with a measure which is greater than or equal to the measure of the original solution.

Let S be a solution to I'. Let v be some variable such that S is not consistent for v. If $\neg v_1 \in S$, then we let $S' = (S \setminus X(v)) \cup O(v)$ where $O(v)$ is the set of nodes contained in an optimal solution to $X(v)$ under the restriction that $\neg v_1 \in O(v)$ (and hence $v_1, v_2 \notin O(v)$). It is clear that S' is an independent set in G as S is an independent set in G.

If $\neg v_1 \notin S$ and ($v_1 \notin S$ or $v_2 \notin S$) we proceed in a similar way. Let $S' = (S \setminus X(v)) \cup O(v)$ where $O(v)$ is as above. In this case S' does not have to be an independent set in G as we have added $\neg v_1$ to S' and a node c adjacent to $\neg v_1$ which corresponds to a clause where $\neg v_1$ participates might be contained in S'. However, by removing this node from S' we get a new set S'' which is an independent set. Furthermore, as S was not consistent for v it follows that the measure of S on $X(v)$ is strictly less than the measure

of S'' on $X(v)$. Hence, the loss we got by removing c is made up for.

We will now show that

$$\text{OPT}(I') \leq |V|K + \text{OPT}(I) \tag{6.2}$$

where $K = 14$ is the optimum value for our gadget. To see this, let S' be an optimal solution to I'. By the argument above we can assume that S' is consistent. Hence, we can construct a solution s to I by letting $s(v) = \text{TRUE}$ if the nodes which corresponds to the literals where the variable v occurs positively is contained in S' and $s(v) = \text{FALSE}$ otherwise. (As S' is consistent if the nodes corresponding to positive occurrences of v are contained in S', then the negative nodes are not contained in S' and vice versa.) Let $l_1 \vee l_2 \in C$ be some constraint, where l_1 and l_2 are literals, and let c and c' denote the two nodes we introduced in G' from this constraint. If $\neg l_1, \neg l_2 \in S'$, then $c, c' \notin S'$ and the constraint is not satisfied in s. On the other hand, if $\neg l_1 \notin S'$ or $\neg l_2 \notin S'$, then either $c \in S'$ or $c \in S'$ and the constraint is satisfied in s. The inequality (6.2) follows from this argument.

As $\text{OPT}(I) \geq |C|/2$ and $|V| \leq 2|C|$ it follows from (6.2) that $\text{OPT}(I') \leq 2K|C| + \text{OPT}(I) \leq (4K+1)\text{OPT}(I)$, hence $\beta = 4K + 1$ is an appropriate parameter for an L-reduction.

For any consistent solution S' to I' we can use the method above to construct a solution s to I (i.e., for each variable $v \in V$ let $s(v) = \text{TRUE}$ if $v_i \notin S'$ and $s(v) = \text{FALSE}$ otherwise). We will then have $|\text{OPT}(I) - m(I, s)| \leq |\text{OPT}(I') - m(I', S')|$. Hence, $\gamma = 1$ is an appropriate parameter for the L-reduction.

Using c_0 and R it is possible to 2-represent $\text{NAND}^2(x, y)$. To reduce I' to an instance of W-MAX ONES($\{R\}$)-2 note that we can "cover" each variable gadget with R and NAND^2, see Figure 6.3 for how this is done. Furthermore, in the covering we have only used v_1, $\neg v_1$ and v_2 once so it will not be any problems with connecting the gadgets to each other with NAND^2 constraints. □

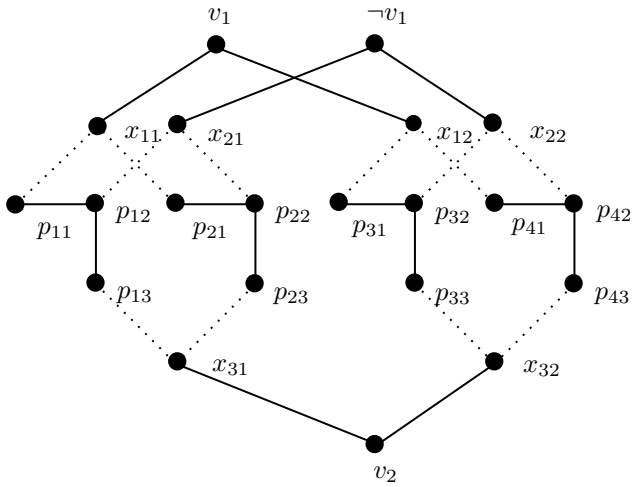
The final subcase is $\Gamma \subseteq IE_2$.

Lemma 6.24. *Given a relation $R \in IE_2$ which is not closed under $f(x, y) = x \vee y$, then R can 2-represent NAND^2.*

Proof. From Lemma 6.8 it follows that R can 2-represent either NAND^2 or $x \neq y$, but the latter is not contained in IE_2, hence we must have the former. □

Lemma 6.25. *Let Γ be a conservative constraint language. If $IS_{12}^2 \subseteq \langle\Gamma\rangle \subseteq IE_2$ and there is a relation $R \in \Gamma$ such that R is not a Δ-matroid relation, then* W-MAX ONES(Γ)-2 *is* **APX**-*hard.*

Some parts of the following proof is similar to the proof of Lemma 6.15. Both of the proofs build upon Feder's work in [56] that non-Δ-matroids causes CSP(Γ)-2 to be no easier than CSP(Γ). This case is a bit more



Part of graph	Constraints
Paths	$R(p_{11}, p_{12}, p_{13}), R(p_{21}, p_{22}, p_{23})$
	$R(p_{31}, p_{32}, p_{33}), R(p_{41}, p_{42}, p_{43})$
Tree for v_1	$R(p_{11}, x_{11}, p_{21}), R(p_{31}, x_{12}, p_{41}), R(x_{11}, v_1, x_{12})$
Tree for $\neg v_1$	$R(p_{12}, x_{21}, p_{22}), R(p_{32}, x_{22}, p_{42}), R(x_{21}, \neg v_1, x_{22})$
Tree for v_2	$R(p_{13}, x_{31}, p_{23}), R(p_{33}, x_{32}, p_{43}), R(x_{31}, v_2, x_{32})$

Figure 6.3: The gadget for the variable v covered by the relation R. Note that each variable occurs at most twice and that v_1, v_2 and $\neg v_1$ occurs once. Constraints with overlapping nodes are represented by two different line styles in the graph: solid and dotted.

complicated compared to Lemma 6.15 because we do not have access to all two-literal clauses here.

Proof. As in Lemma 6.15 we can 2-represent a non-Δ-matroid relation of arity 3 with R, c_0 and c_1. Let P be this relation. As P is not a Δ-matroid relation there exists tuples $\boldsymbol{v}, \boldsymbol{v'} \in P$ such that $d_H(\boldsymbol{v}, \boldsymbol{v'}) = 3$ and a step $\boldsymbol{s} \notin P$ from $\boldsymbol{v}$ to $\boldsymbol{v'}$ such that no step from $\boldsymbol{s}$ to $\boldsymbol{v'}$ is contained in P.

We can assume that $\boldsymbol{v} \oplus 1, \boldsymbol{v} \oplus \{1,2\}, \boldsymbol{v} \oplus \{1,3\} \notin P$. Hence, depending on $\boldsymbol{v}$ and which other tuples that are in P we get a number of possibilities. We will use the following notation: $\boldsymbol{a} = \boldsymbol{v} \oplus 2$, $\boldsymbol{b} = \boldsymbol{v} \oplus 3$ and $\boldsymbol{c} = \boldsymbol{v} \oplus \{2,3\}$. Zero or more of $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ may be contained in P. Tables 6.5–6.8 list the possible relations we can get, up to permutations of the coordinates. Note that $\boldsymbol{a} \in P, \boldsymbol{b}, \boldsymbol{c} \notin P$ and $\boldsymbol{b} \in P, \boldsymbol{a}, \boldsymbol{c} \notin P$ are equivalent if we disregard permutations of the coordinates. Similarly $\boldsymbol{a}, \boldsymbol{c} \in P, \boldsymbol{b} \notin P$ and $\boldsymbol{b}, \boldsymbol{c} \in P, \boldsymbol{a} \notin P$ are equivalent.

We now get three possibilities

1. P is not in IE_2 and can therefore be omitted from further consideration (it is clear that if P is not in IE_2 then R is not in IE_2 either, which is a contradiction with the assumptions in the lemma);
2. P can 2-represent EQ^3, $y \leq x = z$, $y \leq x = z$, or $x \leq y \leq z$; or
3. P can 2-represent $ABC5$.

Table 6.3 lists all the relations and which of the three cases that occur for them.

In Case 2, if we can 2-represent $x = z \leq y$, $y \leq x = z$, or $x \leq y \leq z$, then we can use a circle of constraints to 2-represent EQ^3, as was done in Lemma 6.15. If we can 2-represent EQ^3, we get **poly-APX**-hardness due to the construction in Lemma 6.24, Lemma 6.2 and a simple reduction from MIS.

In Case 3 we can 2-represent the relation $ABC5$, with the representations in Table 6.4. The relation $ABC5$ is equivalent to

$$\mathrm{NAND}^2(x_1, x_2) \wedge \mathrm{NAND}^2(x_1, x_3).$$

By Lemma 6.23 this relation causes W-Max Ones(Γ)-2 to be **APX**-hard. □

The results obtained in Lemma 6.25 is not optimal for all non-Δ-matroids. It is noted in the proof that we get **poly-APX**-hardness results for some of the relations, but we do not get this for all of them. In particular we do not get this for $A3$, $AB1$, $BC4$, $ABC1$, $ABC3$, $ABC5$ and $ABC6$. However, $ABC5$ is contained in **APX** by Lemma 6.23.

6.5 Non-conservative Constraint Languages

In this section we will take a look at the non-conservative case, i.e., we will look at constraint languages which do not necessarily contain c_0 and c_1. A

Relation	2-representation or comment	Relation	2-representation or comment
1	Is EQ^3	BC2	Not in IE_2
2	Not in IE_2	BC3	Not in IE_2
A1	Is $x = z \leq y$	BC4	See Table 6.4
A2	Not in IE_2	AB1	See Table 6.4
A3	See Table 6.4	AB2	Not in IE_2
A4	Not in IE_2	AB3	Not in IE_2
A5	Not in IE_2	AB4	Not in IE_2
A6	Is $y \leq x = z$	AB5	Not in IE_2
C1	Same as A6	AB6	Not in IE_2
C2	Not in IE_2	ABC1	See Table 6.4
C3	Not in IE_2	ABC2	Not in IE_2
C4	Not in IE_2	ABC3	See Table 6.4
C5	Not in IE_2	ABC4	Not in IE_2
C6	Same as A1	ABC5	See Lemma 6.23
BC1	Is $x \leq y \leq z$	ABC6	See Table 6.4

Table 6.3: Non-Δ-matroid relations in Lemma 6.25. The table lists 2-representations of relations which makes W-MAX ONES$(\cdot)$-2 hard for all non-Δ-matroid relations contained in IE_2 of arity three.

Relation	2-representation
A3	$\exists x' : A3(x_1, x_2, x') \wedge \text{NAND}^2(x', x_3)$
BC4	$\exists x' : BC4(x_1, x', x_2) \wedge \text{NAND}^2(x', x_3)$
AB1	$\exists x', x'' : AB1(x_1, x', x'') \wedge$ $\text{NAND}^2(x', x_2) \wedge \text{NAND}^2(x'', x_3)$
ABC1	$\exists x', x'' : ABC1(x_1, x', x'') \wedge$ $\text{NAND}^2(x', x_2) \wedge \text{NAND}^2(x'', x_3)$
ABC3	$\exists x' : ABC3(x_1, x_2, x') \wedge \text{NAND}^2(x', x_3)$
ABC6	$\exists x' : ABC6(x', x_2, x_3) \wedge \text{NAND}^2(x', x_1)$

Table 6.4: 2-representations of $ABC5$. These are used in Lemma 6.25. Due to Lemma 6.24 we are free to use NAND^2 in the representations.

		000	100	010	110	101	111
111	110	111	011	101	001	010	000
000	001	010	110	000	100	111	101
1	2	A1	A2	A3	A4	A5	A6

Table 6.5: Non-Δ-matroid relations where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \notin P$ followed by the relations where $\boldsymbol{a} \in P$ and $\boldsymbol{b}, \boldsymbol{c} \notin P$. The first row is $\boldsymbol{v}$, the second is $\boldsymbol{v} \oplus \{1, 2, 3\}$ and the third is $\boldsymbol{a}$.

000	100	010	110	011	111
111	011	101	001	100	000
011	111	001	101	011	100
C1	C2	C3	C4	C5	C6

Table 6.6: Non-Δ-matroid relations where only $\boldsymbol{c} \in P$. The rows are, from top to bottom, $\boldsymbol{v}$, $\boldsymbol{v} \oplus \{1,2,3\}$, and $\boldsymbol{c}$.

000	100	010	001	000	100	010	110	011	111
111	011	101	110	111	011	101	001	100	000
001	101	011	000	010	110	000	100	001	101
011	111	001	010	001	101	011	111	010	110
BC1	BC2	BC3	BC4	AB1	AB2	AB3	AB4	AB5	AB6

Table 6.7: Non-Δ-matroid relations where $\boldsymbol{b}, \boldsymbol{c} \in P$ and $\boldsymbol{a} \notin P$ followed by relations where $\boldsymbol{a}, \boldsymbol{b} \in P$ and $\boldsymbol{c} \notin P$. In the BC-relations the rows are, from top to bottom, $\boldsymbol{v}, \boldsymbol{v} \oplus \{1,2,3\}$, $\boldsymbol{b}$, and $\boldsymbol{c}$. Similarly, for the AB-relations we have $\boldsymbol{v}$, $\boldsymbol{v} \oplus \{1,2,3\}$, $\boldsymbol{a}$, and $\boldsymbol{b}$.

000	100	010	110	011	111
111	011	101	001	100	000
010	110	000	100	001	101
001	101	011	111	010	110
011	111	001	101	000	100
ABC1	ABC2	ABC3	ABC4	ABC5	ABC6

Table 6.8: Non-Δ-matroid relations where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in P$. The rows are, from top to bottom, $\boldsymbol{v}, \boldsymbol{v} \oplus \{1,2,3\}$, $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$.

relation R is said to be *1-valid* if it contains the all ones tuple, i.e., R is 1-valid if $(1, 1, \ldots, 1) \in R$. A constraint language is said to be 1-valid if every relation in the language is 1-valid.

Theorem 6.26. *For any constraint language* Γ *which is not 1-valid, if* W-MAX ONES($\Gamma \cup \{c_0, c_1\}$)-k *is* **NP**-*hard for some integer* k *then so is* W-MAX ONES(Γ)-k.

Note that for constraint languages Γ which are 1-valid W-MAX ONES(Γ) is trivial: the all-ones solution is optimal. The idea in the proof is that we can simulate c_1 constraints by giving the variable a large weight. Furthermore, if there are relations which are not 1-valid then we can represent c_0 constraints when we have access to c_1 constraints. It fairly easy to see why this fails to give us any inapproximability results: due to the large weight used to simulate c_1 any feasible solution is a good approximate solution.

Proof (Of Theorem 6.26). Let Γ be a non-1-valid constraint language and k an integer such that W-MAX ONES($\Gamma \cup \{c_0, c_1\}$)-k (this problem will hereafter be denoted by Π_{01}) is **NP**-hard. We will prove the theorem with a reduction from Π_{01} to W-MAX ONES(Γ)-k (hereafter denoted by Π).

As Γ is not 1-valid there exists a relation $R \in \Gamma$ such that $(1, \ldots, 1) \notin R$. Let r be the arity of R and let $\boldsymbol{t}$ be the tuple in R with the maximum number of ones. Assume, without loss of generality, that $\boldsymbol{t} = (0, 1, \ldots, 1)$.

The assumption in the theorem implies that it is **NP**-hard to decide the following question: given an instance $I = (V, C, w)$ of Π_{01} and an integer K is $\text{OPT}(I) \geq K$?

Let $I = (V, C, w), K$ be an arbitrary instance of the decision variant of Π_{01}. We will transform I into an instance $I' = (V', C', w'), K'$ of the decision variant of Π by first removing constraint applications using c_0 and then removing constraint applications using c_1.

At the start of the reduction let $V' = V$ and $C' = C$. For each constraint $(c_0, (v)) \in C'$ replace this constraint with $(R, (v, v_1, \ldots, v_{r-1}))$ where $v_1, \ldots, v_{r-1}$ are fresh variables, furthermore add the constraint $(c_1, (v_k))$ for $k = 1, \ldots, r-1$ to C'.

Let c be the number of variables which are involved in c_1 constraints. For each constraint using c_1, $(c_1, (v)) \in C'$, remove this constraint and set $w'(v) = L + w(v)$, where L is a sufficiently large integer ($L = 1 + \sum_{v \in V} w(v)$ is enough). For every variable v which is not involved in a c_1 constraint let $w'(v) = w(v)$.

Finally let $K' = K + cK$. Given a solution s' to I' such that $m(I', s') \geq K'$ it is clear that this solution also is a solution to I such that $m(I, s') \geq K$. Furthermore, if there is a solution s to I such that $m(I, s) \geq K$ then s is a solution I' such that $m(I', s) \geq K'$. □

Part III

Maximum CSP

Chapter 7

Introduction to Max CSP

7.1 Research Directions on Max CSP

In this part of the thesis we will investigate the complexity of Max CSP(Γ) for various constraint languages Γ. One can roughly divide the work done on Max CSP(Γ) into two research directions, which we will describe here.

The first direction is devoted to studying the approximability of Max CSP(Γ). In this line of research a typical hardness result is of the form "if Max CSP(Γ) can be approximated better than c, then something unexpected happens". Here "unexpected" is typically that $\mathbf{P} = \mathbf{NP}$ or that the unique games conjecture (UGC)[1] fails. These results often uses the PCP theorem [8] and the sophisticated theory developed around this characterisation of **NP**.

The algorithmic results are typically of the form "Max CSP(Γ) can be approximated within c in polynomial time" (often randomised polynomial time). These algorithms are almost exclusively based on semi-definite programming (see, e.g., [68, 80, 127]).

The goal of this line of research is to prove *tight* approximability results for every Γ. That is, to prove that for every Γ there is some constant $c(\Gamma)$ such that Max CSP(Γ) can be approximated within $c(\Gamma)$ in polynomial time, but cannot be approximated within $c(\Gamma) - \epsilon$ for any $\epsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$. Of course, one preferably also wants some procedure with which

[1] The UGC is a conjecture due to Koth [103] which states that a certain Max CSP(Γ) problem is hard to approximate in a certain specific sense. Let D be a finite set and say that a binary relation is *unique* if it is of the form $\{(x, \rho(y)) \mid x \in D\}$ for some permutation ρ on D. For a finite set D let Γ_D be the constraint language which contains all unique relations over D. The UGC states that for every $\epsilon > 0$, there is some domain D such that given an instance I of Max CSP(Γ_D) with the promise that either at most an ϵ fraction of the constraints can be satisfied or at least an $1 - \epsilon$ fraction of the constraints can be satisfied, it is **NP**-hard to tell these two cases apart. The UGC is a strengthening of the PCP theorem. It is currently not known if the UGC holds, but there are a number of approximability and inapproximability results which match each other under the assumption that the conjecture is true (see, e.g., [104, 105, 127]).

one can compute $c(\Gamma)$ from Γ.

So far this line of research has concentrated on families of constraint languages such that MAX CSP(Γ) is not in **PO** (unless **P** = **NP**). However, fairly recently there was a breakthrough in this area when Prasad Raghavendra [127] proved almost tight approximability results for every Γ (including the ones for which MAX CSP(Γ) is in **PO**).[2] Raghavendra proved that if the UGC holds, then for every constraint language Γ there is a constant $c(\Gamma)$ such that MAX CSP(Γ) can be approximated within $c(\Gamma) + \epsilon$ for all $\epsilon > 0$ and furthermore that there is no polynomial time approximation algorithm with performance ratio $c(\Gamma) - \epsilon$ for MAX CSP(Γ) for any $\epsilon > 0$ (assuming that $c(\Gamma) > 1$). Note that if the UGC is true and MAX CSP(Γ) is in **PO**, then Raghavendra's result implies a PTAS for MAX CSP(Γ) (but not a polynomial-time algorithm for solving the problem to optimality).

There is also another line of research where one is interested in characterising the constraint languages Γ such that MAX CSP(Γ) can be solved to optimality in polynomial time, that is, when the problem is in **PO**. As far as we know the first systematic study of the latter type of problem was conducted in [35].

A typical hardness result in this line of research is of the form "there is no PTAS for MAX CSP(Γ), unless **P** = **NP**" (or sometimes "MAX CSP(Γ) is **NP**-hard"). Whereas a typical algorithmic result is of the form "MAX CSP(Γ) is in **PO**". In this case the hardness results are based on local substitution in the instances (gadgets) sometimes with a considerable case analysis [50], structure of supermodular predicates [50, 93], or algebraic techniques adapted from the work on CSP (Chapter 12 of this thesis). The algorithmic results comes from the notion of *submodular function minimisation*, which is presented in Section 7.3, or reductions to the minimum cut problem (which can be seen as a submodular function minimisation problem, see Example 7.1). The connection between submodularity and MAX CSP is presented in Section 7.5.

Even though the two lines of research seems to be similar to each other, the techniques used are very different. In this part of the thesis we will present some results which are a part of the latter line of research.

7.2 Lattices

We will need some definitions and notions from lattice theory in the subsequent parts of the thesis. For a general background on lattice theory we refer the reader to [13].

Recall that a partial order $\sqsubseteq$ on a domain L is a *lattice order* if, for every $x, y \in L$, there exists a greatest lower bound $x \sqcap y$ (meet) and a least upper bound $x \sqcup y$ (join). The algebra $\mathcal{L} = (L; \sqcap, \sqcup)$ is a *lattice*, and

[2] In fact Raghavendra's result also applies to VCSP(Γ) for every valued constraint language Γ such that there is no k-ary cost function $f \in \Gamma$ and $\boldsymbol{x} \in D^k$ with $f(\boldsymbol{x}) = -\infty$.

$x \sqcup y = y \iff x \sqcap y = x \iff x \sqsubseteq y$. We will write $x \sqsubset y$ if $x \neq y$ and $x \sqsubseteq y$. If $x \sqsubset y$ and there is no $z \in L$ such that $x \sqsubset z \sqsubset y$ then y *covers* x and we write this as $x \prec y$. We will typically denote the top and bottom elements of a lattice $\mathcal{L}$ by $1_{\mathcal{L}}$ and $0_{\mathcal{L}}$, respectively. The *atoms* of a lattice $\mathcal{L}$ are the elements $x \in L$ such that $0_{\mathcal{L}} \prec x$. Figure 7.1 contains some examples and non-examples of lattices. Unless stated otherwise the lattices we consider will be finite, and we will simply refer to these algebras as *lattices* instead of using the more appropriate term *finite lattices*.

Given any two lattices $\mathcal{A} = (A; \sqcap_A, \sqcup_A)$ and $\mathcal{B} = (B; \sqcap_B, \sqcup_B)$ the *direct product* of $\mathcal{A}$ and $\mathcal{B}$ is denoted by $\mathcal{A} \times \mathcal{B}$ and is the lattice with domain $A \times B$ and operations $\sqcap, \sqcup$ defined by $(x_A, x_B) \sqcap (y_A, y_B) = (x_A \sqcap_A y_A, x_B \sqcap_B y_B)$ and $(x_A, x_B) \sqcup (y_A, y_B) = (x_A \sqcup_A y_A, x_B \sqcup_B y_B)$ for any $x_A, y_A \in A$ and $y_A, y_B \in B$. The *direct power* of $\mathcal{L}$, denoted by $\mathcal{L}^n$, is the direct product

$$\underbrace{\mathcal{L} \times \mathcal{L} \times \ldots \times \mathcal{L}}_{n \text{ times}}.$$

(So it is the lattice with domain L^n and operations acting componentwise.) The top and bottom of $\mathcal{L}^n$ will be denoted by $\mathbf{1}_{\mathcal{L}^n}$ and $\mathbf{0}_{\mathcal{L}^n}$, respectively. For an arbitrary tuple $\boldsymbol{t} \in \mathcal{L}^n$, index $i \in [n]$, and element $x \in \mathcal{L}$ we use the notation $\boldsymbol{t}[i = x]$ to denote the tuple which is identical to $\boldsymbol{t}$ except possibly in coordinate i where $\boldsymbol{t}[i = x]$ equals x. Sometimes this notation is extended to multiple coordinates, e.g., $\boldsymbol{t}[i = x, j = y]$ denotes a tuple which is identical to $\boldsymbol{t}$ except possibly in coordinates i and j, where it takes the values x and y, respectively (when this notation is used i and j will always be distinct).

If $\mathcal{L}$ is a lattice and S is a subset of L such that for any $x, y \in S$ we have $x \sqcap y, x \sqcup y \in S$, then $\mathcal{S} = (S; \sqcap, \sqcup)$ is a *sublattice of* $\mathcal{L}$. From now on we will often drop the distinction between a lattice and its domain, so we will write things like "$x \sqcap y, x \sqcup y \in S$ and thus S is a sublattice of L" with the intended meaning "$x \sqcap y, x \sqcup y \in S$ and thus $\mathcal{S}$ is a sublattice of $\mathcal{L}$".

A lattice $\mathcal{L}$ is *distributive* if

$$x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$$

for all $x, y, z \in L$. Given any finite set V the sets of subsets of V, 2^V, is a lattice under the $\subseteq$ order. In this lattice meet is given by the intersection of two sets and join is given by union. By Birkhoff's representation theorem [13] a lattice is distributive if and only if it is isomorphic to a sublattice of 2^V for some set V. It is well-known that distributive lattices also can be characterised as the lattices which neither have the five element diamond nor the pentagon as sublattices [13] (see Figure 7.1 for diagrams of these lattices).

A lattice $\mathcal{L}$ is *modular* if

$$x \sqsubseteq z \Rightarrow x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap z$$

for all $x, y, z \in L$. There is another equivalent definition of modular lattices which sometimes is easier to work with. To state the second definition

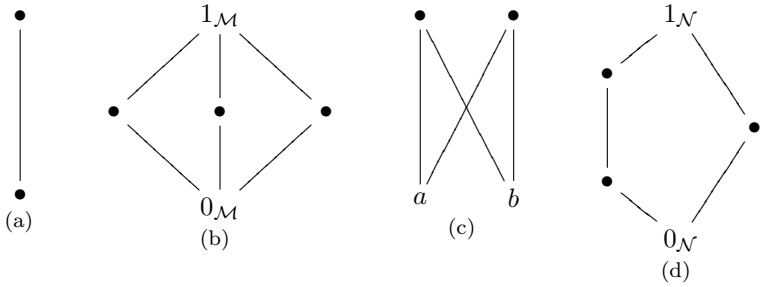


Figure 7.1: The figure depicts the Hasse diagrams of four posets. (a) is the unique two element lattice. (b) is the five element lattice $\mathcal{M}$ called the *diamond*. (c) is not a lattice, because a and b does not have a least upper bound. (d) is a lattice called the *pentagon*. (a) is the only distributive lattice in this figure and (a) and (b) are the two modular lattices in the figure.

we need to introduce rank functions. A *rank function* for $\mathcal{L}$ is a function $\rho : L \to \mathbb{N}$ such that if $x, y \in L$, $x \prec y$ then $\rho(y) = \rho(x) + 1$. We also require that $\rho(0_L) = 0$. The second characterisation of modular lattices is that a lattice $\mathcal{L}$ is modular if and only if there is a modular rank function ρ for $\mathcal{L}$, that is $\rho(x) + \rho(y) = \rho(x \sqcup y) + \rho(x \sqcap y)$ for all $x, y \in L$.

A third characterisation of modular lattices is that a lattice is modular if and only if it does not have the pentagon (see Figure 7.1) as a sublattice. By this third definition it is clear that the class of all distributive lattices is strictly contained in the class of all modular lattices (in particular the diamond is modular but not distributive).

The distributive lattices and the modular lattices are two classes of lattices which are closed under taking sublattices and direct products (and also homomorphic images, these will be defined in Chapter 10).

7.3 Submodularity

The notion of submodular functions and the problem of minimising such functions turns out to be important for the complexity of MAX CSP. In this section we will define submodular functions and define the minimisation problem. In Section 7.5 the connection between submodular function minimisation and MAX CSP is made precise.

Let V be some finite set. A function $f : 2^V \to \mathbb{R}$ is said to be *submodular* if

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$$

for all $A, B \subseteq V$. If $-f$ is submodular, then f is said to be *supermodular*. In the sequel we will call such functions *submodular set functions* (and *supermodular set functions*, respectively). Submodular set functions show up in various fields including combinatorial optimisation, graph theory [62], game

theory [135], information theory [63, 74] and statistical physics [2]. Examples include the cut functions of graphs and the rank function of matroids. There is also a connection between submodular function minimisation and convex optimisation. In particular, submodularity can be seen as a discrete analogue of convexity [64, 112]. We refer the reader to [65, 86, 112, 116] for a general background on submodular set functions.

Given a submodular set function $f : 2^V \to \mathbb{R}$ there are several algorithms for finding minimisers of f, i.e., finding a subset $X \subseteq V$ such that $f(X) = \min_{Y \subseteq V} f(Y)$, in time polynomial in $|V|$.[3] The first algorithm for finding such minimisers in polynomial-time is due to Grötschel et al. [72]. However, this algorithm is based on the Ellipsoid algorithm and hence its usefulness in practise is limited. Almost two decades later two combinatorial algorithms were found independently by Schrijver [133] and Iwata et al. [87]. More recently the running times have been improved. The currently fastest strongly polynomial time algorithm is due to Orlin [120] and the fastest weakly polynomial time algorithm is due to Iwata [85]. (An algorithm is said to run in *strongly polynomial time* if the running time of the algorithm is bounded by a polynomial in the dimension of the input and is independent of the actual numeric values in the input. In this model the basic arithmetic operations take a unit time step to compute, regardless of the sizes of the operands. An algorithm running in *weakly polynomial time* runs in polynomial time measured in the size of the input, but it is not necessarily strongly polynomial. The algorithm due to Orlin mentioned above runs in time $O(n^5 EO + n^6)$ and the one due to Iwata runs in time $O((n^4 EO + n^5) \log M)$. If $f : 2^V \to \mathbb{Z}$ is the submodular function to be minimised, then $n = |V|$, EO is the time taken to evaluate the oracle function f, and M is the maximum absolute value of f.)

As mentioned above the cut function of graphs is submodular.

Example 7.1 (The Cut Function)**.** *Let $G = (V, E)$ be a graph and let s and t be two distinct vertices in G. The* s,t-cut function *of G is defined by $c : 2^{V \setminus \{s,t\}} \to \mathbb{N}$ and for a subset X of $V \setminus \{s,t\}$ we have that $c(X)$ is the number of edges with exactly one vertex in $X \cup \{s\}$. Minimising the cut function of graphs is a well-known combinatorial optimisation problem.*

It is not hard to see that the cut function is submodular. Indeed, let A and B be two arbitrary subsets of $V \setminus \{s,t\}$. If some edge has exactly one vertex in $A \cup B \cup \{s\}$, then this edge has exactly one vertex in $A \cup \{s\}$ or $B \cup \{s\}$. Similarly, if some edge has exactly one edge in $(A \cap B) \cup \{s\}$, then this edge has exactly one edge in $A \cup \{s\}$ or $B \cup \{s\}$. We have now shown that the inequality

$$c(A \cup B) + c(A \cap B) \leq c(A) + c(B)$$

[3]There is a question of representability of the function f in these results. How is f given? We cannot give $f(X)$ for all $X \subseteq V$ to the algorithm as this is an exponentially large data set (measured in $|V|$). This is solved by providing the algorithm with a value-giving oracle for computing f. That is, the algorithm is given as input a procedure for computing a function from 2^V to $\mathbb{R}$ and this function is assumed to be submodular.

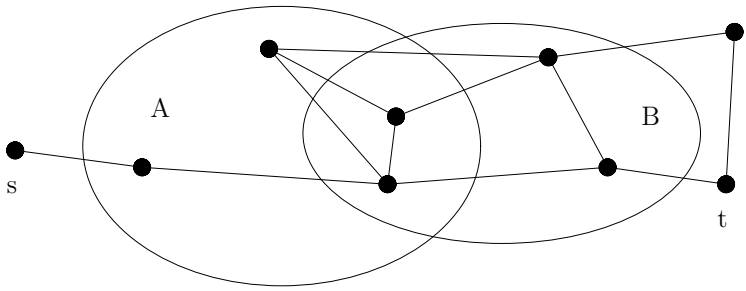


Figure 7.2: A graph and its cut function on two subsets of the vertices. In this case $c(A) = 3$, $c(B) = 7$, $c(A \cup B) = 2$ and $c(A \cap B) = 6$. So $8 = c(A \cup B) + c(A \cap B) \leq c(A) + c(B) = 10$.

holds for all $A, B \subseteq V \setminus \{s, t\}$. See Figure 7.2 for an example graph. Hence, the problem of minimising submodular set functions generalises the problem of finding minimum cuts in graphs.

There is a more general notion of submodularity based on finite lattices. Given a lattice $\mathcal{L}$ we say that a function $h : L^n \to \mathbb{R}$ is submodular if

$$h(\boldsymbol{a} \sqcap \boldsymbol{b}) + h(\boldsymbol{a} \sqcup \boldsymbol{b}) \leq h(\boldsymbol{a}) + h(\boldsymbol{b})$$

for all $\boldsymbol{a}, \boldsymbol{b} \in L^n$. Note that the subsets of V can be seen as a lattice with union as join and intersection as meet (this lattice is a product of the two element lattice). Hence, this notion of submodularity is a generalisation of submodular set functions. In [109] the idea of generalising submodular functions to a direct product of some finite lattice was taken one step further, instead of requiring that the domain of f is a direct product of a single finite lattice the function is defined over a direct product of lattices taken from a finite class of finite lattices. Formally the problem is defined as follows.

Definition 7.2 (SFM($\mathbf{C}$) [109])**.** *For a finite class of finite lattices $\mathbf{C}$, SFM($\mathbf{C}$) is a minimisation problem with*

Instance: *A positive integer n and a submodular function $f : \mathcal{L} \to \mathbb{Z}$ where*

$$\mathcal{L} = \mathcal{L}_1 \times \mathcal{L}_2 \times \ldots \times \mathcal{L}_n$$

and for each $i \in [n]$ we have $\mathcal{L}_i \in \mathbf{C}$ (f is provided to the algorithm as a value-giving oracle).

Solution: *A tuple $\boldsymbol{x} \in \mathcal{L}$.*

Measure: $f(\boldsymbol{x})$

For a single lattice $\mathcal{L}'$ we will write SFM($\mathcal{L}'$) for the problem SFM($\{\mathcal{L}'\}$).

7.4 Three Notions of Non-hardness for SFM

In this section we will give definitions of three notions of non-hardness for SFM($\mathbf{C}$).[4] Let n be a positive integer and let the lattice $\mathcal{L}$ be defined by

$$\mathcal{L}_1 \times \mathcal{L}_2 \times \ldots \times \mathcal{L}_n$$

where $\mathcal{L}_i \in \mathbf{C}$ for each $i \in [n]$. For a function $f : \mathcal{L} \to \mathbb{R}$ we will denote the quantity $\max_{\boldsymbol{t} \in \mathcal{L}} |f(\boldsymbol{t})|$ by $\max(|f|)$. In the definitions below, and in all non-hardness results, we assume that all the submodular functions $f : \mathcal{L} \to \mathbb{Z}$ that appear satisfies $\log \max(|f|) \in O(n^c)$ for some fixed constant c. Without this restriction the algorithms cannot even evaluate f in polynomial time (measured in n). The function is "given" to the algorithms by an oracle which given a tuple $\boldsymbol{x} \in \mathcal{L}$ computes the value $f(\boldsymbol{x})$ in a single time step. Furthermore, the algorithms can assume that the oracle actually computes a submodular function (and not some function which is not submodular).

The notion of *oracle-tractability* for a class of lattices was introduced in [109]. Formally it is defined as follows.

Definition 7.3 (Oracle-tractable [109])**.** *A class of lattices* $\mathbf{C}$ *is* oracle-tractable *if for every finite subclass* $\mathbf{C}'$ *of* $\mathbf{C}$ *there is some* $c \in \mathbb{N}$ *such that given* $n \in \mathbb{N}$ *and an oracle for computing a submodular function* $f : \mathcal{L} \to \mathbb{Z}$*, where*

$$\mathcal{L} = \mathcal{L}_1 \times \mathcal{L}_2 \times \ldots \times \mathcal{L}_n$$

and $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n \in \mathbf{C}'$*, a minimiser of* f *can be found in time* $O(n^c)$*.*

In addition to oracle-tractability we define two other notions of non-hardness.

Definition 7.4 (Oracle-pseudo-tractable)**.** *A class of lattices* $\mathbf{C}$ *is* oracle-pseudo-tractable *if for every finite subclass* $\mathbf{C}'$ *of* $\mathbf{C}$ *there is some* $c \in \mathbb{N}$ *such that given* $n \in \mathbb{N}$ *and an oracle for computing a submodular function* $f : \mathcal{L} \to \mathbb{Z}$*, where*

$$\mathcal{L} = \mathcal{L}_1 \times \mathcal{L}_2 \times \ldots \times \mathcal{L}_n$$

and $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n \in \mathbf{C}'$*, a minimiser of* f *can be found in time* $O(n^c \cdot \max(|f|)^c)$*.*

Definition 7.5 (Well-characterised)**.** *A class of lattices* $\mathbf{C}$ *is* well-characterised *if for every finite subclass* $\mathbf{C}'$ *of* $\mathbf{C}$ *there is some* $c \in \mathbb{N}$ *and a polynomial-time verification algorithm* V *such that given* $m \in \mathbb{Z}$*,* $n \in \mathbb{N}$*, and an oracle for computing a submodular function* $f : \mathcal{L} \to \mathbb{Z}$*, where*

$$\mathcal{L} = \mathcal{L}_1 \times \mathcal{L}_2 \times \ldots \times \mathcal{L}_n$$

and $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n \in \mathbf{C}'$*, then*

[4] We do not want to say "three notions of tractability" as one can hardly say that a problem is tractable just because it is well-characterised.

- *if* $m = \min_{\boldsymbol{y} \in \mathcal{L}} f(\boldsymbol{y})$*, there is a proof* $\boldsymbol{x}$ *of length* $O(n^c)$ *such that* V *accepts on input* $(m, n, \boldsymbol{x})$*;*
- *otherwise, there is no* $\boldsymbol{x}$ *such that* V *accepts on input* $(m, n, \boldsymbol{x})$*.*

A priori it is clear that SFM($\mathcal{L}$) is in the appropriately modified variant of **NP**. That is, for any submodular $f : \mathcal{L}^n \to \mathbb{Z}$ such that $\min_{\boldsymbol{t} \in \mathcal{L}^n} f(\boldsymbol{t}) = m$ there are proofs that can be checked in polynomial time of the fact that $\min_{\boldsymbol{t} \in L^n} f(\boldsymbol{t}) \leq m$. Such a proof simply consists of a tuple $\boldsymbol{t} \in \mathcal{L}^n$ such that $f(\boldsymbol{t}) \leq m$. Well-characterisedness corresponds to the notion of being contained in **coNP**. If a lattice $\mathcal{L}$ is well-characterised, then there are proofs that $\min_{\boldsymbol{t} \in \mathcal{L}^n} f(\boldsymbol{t}) \geq m$ which can be checked in time polynomial in n. It is *not* clear a priori that all lattices are well-characterised.

There is an order among these notions of non-hardness. Oracle-tractability implies oracle-pseudo-tractability and well-characterisedness. On the other hand, it is *not* known if oracle-pseudo-tractability implies well-characterisedness. Neither can we say that well-characterisedness implies oracle-pseudo-tractability.

These definitions naturally leads to the following questions: for which finite lattices $\mathcal{L}$ is SFM($\mathcal{L}$) oracle-tractable (oracle-pseudo-tractable, well-characterised)? The first question, about oracle-tractability, was as far as we know first asked by Cohen et al. [35].

Some lattices are known to be oracle-tractable. In particular, Schrijver [133] showed that given a sublattice S of 2^V (i.e., $S \subseteq 2^V$ and for any $X, Y \in S$ we have $X \cap Y, X \cup Y \in S$) and submodular function $f : S \to \mathbb{R}$ a minimiser of f can be found in time polynomial in $|V|$. Schrijver's result implies that for any distributive lattice $\mathcal{L}$ the problem SFM($\mathcal{L}$) is oracle-tractable.

Krokhin and Larose [109] showed that certain constructions on lattices preserve oracle-tractability of SFM. Their results are of the form "if the lattice $\mathcal{L}$ is oracle-tractable, then so is $\mathcal{L}'$" where the lattice $\mathcal{L}'$ is obtained from $\mathcal{L}$ by certain constructions. We will describe their results in more detail in Chapter 10. One consequence of their results is that the pentagon is oracle-tractable. On the other hand, Krokhin and Larose also showed the diamond is *not* covered by their constructions. In Chapter 8 we will show that the diamond is well-characterised and oracle-pseudo-tractable. In Chapter 9 we will show that a certain superset of the modular lattices are well-characterised. Finally in Chapter 10 we will see how some constructions on lattices give rise to new non-hardness results for SFM.

One construction which is fundamental is taking the union of two classes of lattices which are oracle-tractable (oracle-pseudo-tractable, well-characterised). This lemma will be used in Chapter 9 and Chapter 10. (This lemma is actually a special case of the Mal'tsev product construction introduced in this context in [109]. As mentioned above this construction will be defined in Chapter 10.)

Lemma 7.6. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *are oracle-tractable (oracle-pseudo-tractable, well-characterised) classes of lattices, then so is* $\mathbf{A} \cup \mathbf{B}$.

Proof. We prove the lemma for the case when $\mathbf{A} = \{\mathcal{A}\}$ and $\mathbf{B} = \{\mathcal{B}\}$. The proof easily generalises to arbitrary classes of lattices. Furthermore, we only show the lemma for oracle-tractability, the result for oracle-pseudo-tractability and well-characterisedness is shown in the same way. Let $f : \mathcal{L} \to \mathbb{R}$ be submodular where $\mathcal{L} = \mathcal{A}^n \times \mathcal{B}^m$ for some positive integers n and m. We define a function $g : \mathcal{A}^n \to \mathbb{R}$ as follows

$$g(\boldsymbol{x}) = \min_{\boldsymbol{y} \in \mathcal{B}^m} f(\boldsymbol{x}, \boldsymbol{y}).$$

We claim that g is submodular. To show this let $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{A}^n$ be two arbitrary tuples and choose $\boldsymbol{a'}, \boldsymbol{b'} \in \mathcal{B}^m$ so that $g(\boldsymbol{a}) = f(\boldsymbol{a}, \boldsymbol{a'})$ and $g(\boldsymbol{b}) = f(\boldsymbol{b}, \boldsymbol{b'})$. We now get

$$\begin{aligned} g(\boldsymbol{a}) + g(\boldsymbol{b}) &= f(\boldsymbol{a}, \boldsymbol{a'}) + f(\boldsymbol{b}, \boldsymbol{b'}) \\ &\geq f(\boldsymbol{a} \sqcup \boldsymbol{b}, \boldsymbol{a'} \sqcup \boldsymbol{b'}) + f(\boldsymbol{a} \sqcap \boldsymbol{b}, \boldsymbol{a'} \sqcap \boldsymbol{b'}) \\ &\geq g(\boldsymbol{a} \sqcup \boldsymbol{b}) + g(\boldsymbol{a} \sqcap \boldsymbol{b}). \end{aligned}$$

Here the first equality follows from our choice of $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{a'}$, and $\boldsymbol{b'}$. The first inequality is the submodularity of f and the final inequality follows from the definition of g.

As g is submodular on $\mathcal{A}$ it follows that we can minimise g in time polynomial in m if we can evaluate g for each $\boldsymbol{x} \in \mathcal{A}^n$. For each $\boldsymbol{x} \in \mathcal{A}^n$ the evaluation of $g(\boldsymbol{x})$ is a submodular function minimisation problem over the lattice $\mathcal{B}^m$ and as $\mathcal{B}$ is oracle-tractable this evaluation can be done in time polynomial in m. It is not hard to see that the minimum of f coincides with the minimum of g. Hence, to minimise f it is sufficient to minimise g. We conclude that $\mathbf{A} \cup \mathbf{B} = \{\mathcal{A}, \mathcal{B}\}$ is oracle-tractable. □

7.5 MAX CSP and Supermodularity

There is a connection between submodularity and MAX CSP(Γ) in the following way: recall that we can see the relations in Γ as predicates, that is, functions mapping f mapping D^m to $\{0, 1\}$ (here m is the arity of the relation or function). The problem is then to maximise a sum of the form

$$\sum_{i=1}^{n} f_i(\boldsymbol{x_i}) \tag{7.1}$$

where each $\boldsymbol{x_i}$ contains a subset of the variables in the MAX CSP(Γ) instance and $f_i \in \Gamma$ for each i. If there is some lattice ordering, $\mathcal{L} = (D; \sqcap, \sqcup)$, such that each $f \in \Gamma$ is supermodular on $\mathcal{L}$ then the sum (7.1) is supermodular on $\mathcal{L}$ as well. (The sum of two supermodular functions is supermodular.)

Hence, if each function in Γ is supermodular, then MAX CSP(Γ) reduces to maximise a supermodular function. As $-f$ is supermodular if and only if f is submodular, it follows that maximising a supermodular function is the same as minimising the corresponding submodular function. We will say that a constraint language $\Gamma \subseteq R_D$ is *supermodular on* $\mathcal{L}$ if $\mathcal{L}$ is a lattice ordering of D such that each $f \in \Gamma$ is supermodular on $\mathcal{L}$.

This connection between submodular function minimisation and MAX CSP(Γ) was first observed in [35] and in latter papers the connection was explored further [50, 93, 94, 95, 109]. In particular this means that proving oracle-tractability for new lattices for the SFM problem implies tractability results (solvability in polynomial time) for constraint language restrictions of MAX CSP. Providing good characterisations of SFM($\mathcal{L}$), as we do in Chapter 8 and 9, implies **coNP** containment results for the appropriate restriction of MAX CSP. As MAX CSP is trivially in **NP** we get containment in **NP** $\cap$ **coNP** for these restrictions. We have collected these implications in Theorem 7.7 below.

To state the conjectured dichotomy for MAX CSP we need to introduce *cores*. A *retraction* π of a constraint language $\Gamma \subseteq R_D$ is a unary polymorphism of Γ such that if D' is the image of π, then $\pi(x) = x$ for all $x \in D'$. If all retractions of Γ are injective, then Γ is a core. So if Γ is not a core, then there is some unary polymorphism $\pi : D \to D$ which is not injective. In this case MAX CSP(Γ) has the same complexity as MAX CSP($\pi(\Gamma)$) where $\pi(\Gamma) = \{\pi(R) \mid R \in \Gamma\}$. What happens here is that from any solution s to MAX CSP(Γ) the assignment $\pi \circ s$ is a solution to MAX CSP(Γ) with the same measure. Hence, some of the values in D are "useless" in the sense that they can be avoided without losing anything. We remark that the analogous result holds for the complexity of CSP(Γ): CSP(Γ) is polynomial time equivalent to CSP($\pi(\Gamma)$) [76]. If we choose π so that $|D'|$ is minimal, then $\pi(\Gamma) = \Gamma|_{D'}$ is a core of Γ. As in the case of graphs, all cores of Γ are isomorphic, so one can speak about *the* core of Γ. [76]

Several partial results about the complexity of MAX CSP(Γ) have been established. In particular for the cases below a complete characterisation of the complexity of MAX CSP(Γ) is known.

- Two element domains ([42] and [44]);
- three element domains ([93]);
- constraint languages containing all unary constraints ([50]); and
- constraint languages consisting of one relation ([94] and Chapter 12 of this thesis).

In every case the result is that if the core of Γ is supermodular on some lattice ordering of the domain, then MAX CSP(Γ) is tractable and otherwise it is **NP**-hard. (Often stronger hardness results have been achieved, where PTASes are excluded unless **P** = **NP** and/or the instances can be restricted

to have a bounded number of occurrences of each variable. We prove such a result for constraint languages consisting of a single relation in Chapter 12.)

It has been conjectured that supermodularity characterises all tractable cases of MAX CSP. [35] This conjecture is open on both ends: it is not known that all constraint languages which are supermodular on some lattice ordering on the domain are tractable and neither is it known that all constraint languages which are not supermodular on any lattice ordering give rise to non-tractable problems.

We are now ready to present the connections between the different notions of non-hardness for SFM and the complexity of MAX CSP.

Theorem 7.7. *Let $\Gamma \subseteq R_D$ be a constraint language which is supermodular on some lattice $\mathcal{L} = (D; \sqcup, \sqcap)$.*

1. *If SFM($\mathcal{L}$) is oracle-tractable, then* W-MAX CSP(Γ) *is in* **PO**;
2. *if SFM($\mathcal{L}$) is oracle-pseudo-tractable, then*
   (a) MAX CSP(Γ) *is in* **PO**, *and*
   (b) W-MAX CSP(Γ) *is solvable in pseudopolynomial time;*
3. *if SFM($\mathcal{L}$) is well-characterised, then* W-MAX CSP(Γ) *is in* **NP** $\cap$ **coNP**.

Proof. Given an instance I of (W-)MAX CSP(Γ) with n variables we can construct a supermodular function $f : D^n \to \mathbb{N}$ such that for any $\boldsymbol{x} \in D^n$ the measure of $\boldsymbol{x}$ on I equals $f(\boldsymbol{x})$. As f is supermodular $-f$ is submodular. Case 1 now follows from the oracle-tractability of SFM($\mathcal{L}$). In Case 2 we can minimise $-f$ in time polynomial in n and $\max(|f|)$. In Case 2a, as I is unweighted we have $\max(|f|) = \max(f) = \text{OPT}(I) \leq |I|$. Hence, we can minimise $-f$ in time polynomial in $|I|$. As for Case 2b, if $I = (V, C, w)$ then

$$\max(|f|) \leq \sum_{v \in V} w(v)$$

so $\max(|f|)$ is bounded by a polynomial in the weights of the instance. Hence,W-MAX CSP(Γ) can be solved in pseudopolynomial time.

Finally, Case 3 follows from the fact that MAX CSP(Γ) is in **NP** for all Γ and the well-characterisedness of SFM($\mathcal{L}$). □

7.6 VCSP and Supermodularity

Supermodular function maximisation can also be used to prove tractability for certain valued constraint languages for VCSP. In particular, it is straightforward to extend Theorem 7.7 to the VCSP case. We state the latter result as Theorem 7.8 below.

Let Γ be a valued constraint language over a domain D such that for any $f \in \Gamma$ of arity n_f and any $\boldsymbol{x} \in D^n$ we have $f(\boldsymbol{x}) \neq -\infty$. Similar to the

definition in Section 7.5 we say that Γ *supermodular on* $\mathcal{L}$ if $\mathcal{L}$ is a lattice with domain D such that each cost function $f \in \Gamma$ is supermodular on $\mathcal{L}^{n_f}$, where n_f is the arity of f.

Theorem 7.8. *Let* Γ *be a valued constraint language on the domain* D *such that*

- *for any* n_f*-ary* $f \in \Gamma$ *and any* $\boldsymbol{x} \in D^{n_f}$ *we have* $f(\boldsymbol{x}) \neq -\infty$*, and*
- Γ *is supermodular on some lattice* $\mathcal{L} = (D; \sqcup, \sqcap)$*.*

Then,

1. *if SFM*$(\mathcal{L})$ *is oracle-pseudo-tractable, then* VCSP(Γ) *is in* **PO***;*
2. *if SFM*$(\mathcal{L})$ *is well-characterised, then* VCSP(Γ) *is in* **NP** $\cap$ **coNP**.

As we have not introduced a weighted variant of VCSP this theorem has fewer cases than Theorem 7.8. The theorem above can be proved in the same way as Theorem 7.8, we omit the details. The restriction that no cost function in Γ maps any tuple to $-\infty$ can be lifted if it is possible to minimise submodular functions defined on sublattices of $\mathcal{L}^n$ as well. We will derive a result of this kind in Section 10.2 for modular lattices. We will not elaborate further on the connection between VCSP and sublattices here.

Chapter 8

SFM on Diamonds

8.1 Introduction

A lattice $\mathcal{L}$ is a *diamond* if the elements of the lattice form a disjoint union of $\{0_\mathcal{L}, 1_\mathcal{L}\}$ and A, for some finite set A such that $|A| \geq 3$. Here $0_\mathcal{L}$ is the bottom element of $\mathcal{L}$, and $1_\mathcal{L}$ is the top element of $\mathcal{L}$, and all elements in A are incomparable to each other. Figure 7.1b depicts the diamond with three atoms. We will denote the diamond with k atoms by $\mathcal{M}_k$. In this chapter the complexity of SFM($\mathcal{M}_k$) is investigated for $k \geq 3$.

Before this work the five element diamond was the smallest lattice (minimal number of elements) for which no non-hardness results was known for SFM. (See Section 7.4 for a description of the known results in this area.) In this chapter we prove

1. a min–max theorem, which states that the minimum of the submodular function is equal to the maximum of a certain function defined over a certain polyhedron; and
2. that $\mathcal{M}_k$ is well-characterised for all $k \geq 3$; and
3. that $\mathcal{M}_k$ is oracle-pseudo-tractable for all $k \geq 3$.

The first result in this chapter, the min–max theorem for SFM($\mathcal{M}_k$), is stated as Theorem 8.5 and Theorem 8.7. This result looks quite similar to Edmonds' min–max theorem for submodular set functions [53] (we present Edmonds' result in Section 8.2). The key step in the proof of this result is the definition of a certain polyhedron, which depends on f.

The second result is the well-characterisedness of $\mathcal{M}_k$. The proof of this result makes use of Carathéodory's theorem and of the known polynomial-time algorithms for minimising submodular set functions. We also need our min–max theorem.

The third result is the oracle-pseudo-tractability of $\mathcal{M}_k$. The algorithm for this result is presented in Section 8.7. The main part of the algorithm

consists of a nested application of the Ellipsoid algorithm. We also need to prove that the polyhedrons we associate with submodular functions are 1/2-integral. It is still an open problem if $\mathcal{M}_k$ is oracle-tractable for any $k \geq 3$ (we conjecture that it is).

As discussed in Chapter 7 one can prove tractability for certain constraint languages for Max CSP and VCSP by proving that lattices are (pseudo-)oracle-tractable. There are some known results for the complexity of Max CSP(Γ) when Γ is supermodular on a diamond. In particular, Krokhin and Larose [109] showed that W-Max CSP(Γ) can be solved in polynomial time in this case (they did this by a clever reduction to the weighted minimum cut problem in directed graphs, which is known to be solvable in polynomial time). This means that the results in this chapter do not directly yield new complexity theoretic results for Max CSP. However, in Chapter 10 we will present some constructions on lattices which can be used to prove non-hardness results for new lattices. By combining the results in this chapter with these constructions we do indeed get new tractability results for Max CSP. We note that these constructions do currently require that one can minimise arbitrary submodular functions over the diamonds, in particular it is not sufficient to restrict oneself to submodular functions which can be expressed as a sum of predicates. Note that the objective function of Max CSP(Γ) when Γ is supermodular on some lattice is a sum of supermodular predicates. Hence, the tractability of Max CSP(Γ) when Γ is supermodular on some diamond is not sufficient for the constructions we will work with in Chapter 10. For VCSP the situation is different as Krokhin and Larose's result does not apply to VCSP. Hence, in the VCSP case the results in this chapter implies new tractability results (they follow from Theorem 7.8 and the results in this chapter).

This chapter is organised as follows, in Section 8.2 we give a short background on submodular set functions, in Section 8.3 we introduce some more notation which will be used in this chapter and in Chapter 9, in Section 8.4 we prove the first result—the min–max theorem. In Section 8.5 we state some definitions and results which are related to the Ellipsoid algorithm, which we need in our algorithms. The well-characterisation theorem is given in Section 8.6. In Section 8.7 the pseudopolynomial-time algorithm for the minimisation problem is described and proved correct.

8.2 Background on Submodular Set Functions

In this section we will give a short background on Edmonds' min–max theorem for submodular set functions. This result was first proved by Edmonds in [53], but see also the surveys [86, 116].

Let V be a finite set. For a vector $\boldsymbol{x} \in \mathbb{R}^V$ (i.e., $\boldsymbol{x}$ is a function from V into $\mathbb{R}$) and a subset $Y \subseteq V$ define $\boldsymbol{x}(Y) = \sum_{y \in Y} \boldsymbol{x}(y)$. We write $\boldsymbol{x} \leq 0$ if $\boldsymbol{x}(v) \leq 0$ for all $v \in V$ and $\boldsymbol{x}^-$ for the vector in which coordinate v has the value $\min\{0, \boldsymbol{x}(v)\}$. Let f be a submodular set function $f : 2^V \to \mathbb{R}$ such

that $f(\emptyset) = 0$ (this is not really a restriction, given a submodular function g we can define a new function $g'(X) = g(X) - g(\emptyset)$, g' satisfies $g'(\emptyset) = 0$ and is submodular). The *submodular polyhedron* and the *base polyhedron* defined by

$$P(f) = \{\boldsymbol{x} \in \mathbb{R}^V \mid \forall Y \subseteq V, \boldsymbol{x}(Y) \leq f(Y)\}, \text{ and}$$
$$B(f) = \{\boldsymbol{x} \in \mathbb{R}^V \mid \boldsymbol{x} \in P(f), \boldsymbol{x}(V) = f(V)\}$$

often play an important role in results related to submodular set functions. Edmonds [53] proved the following min–max theorem

$$\begin{aligned} \min_{X \subseteq V} f(X) &= \max\{\boldsymbol{x}(V) \mid \boldsymbol{x} \in P(f), \boldsymbol{x} \leq 0\} \\ &= \max\{\boldsymbol{x}^-(V) \mid \boldsymbol{x} \in B(f)\}. \end{aligned} \tag{8.1}$$

In Section 8.4 we give an analogue to (8.1) for submodular functions over diamonds.

8.3 Preliminaries

Recall that the diamonds are modular lattices (the rank function ρ is defined by $\rho(0_{\mathcal{M}}) = 0$, $\rho(a) = 1$ for all $a \in A$ and $\rho(1_{\mathcal{M}}) = 2$). As direct products of modular lattices also are modular lattices it follows that direct products of diamonds are modular lattices.

Throughout this chapter n will denote a positive integer. For an integer $i \in \{0, 1, \ldots, n\}$ we will use $\boldsymbol{v_i}$ to denote the tuple defined by $\boldsymbol{v_i}(j) = 1_{\mathcal{M}}$ for all $j \in [i]$ and $\boldsymbol{v_i}(j) = 0_{\mathcal{M}}$ otherwise. (So, in particular, $\boldsymbol{v_0} = \boldsymbol{0}_{\boldsymbol{\mathcal{M}^n}}$.)

For a set X we let $\mathbb{R}^{[n]\times X}$ be the set of functions mapping $[n] \times X$ into $\mathbb{R}$. Such functions will be called *vectors* and can be seen as vectors indexed by pairs from $[n] \times X$. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{[n]\times X}$ and $\alpha \in \mathbb{R}$ we define $\alpha\boldsymbol{x}, \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{x}^- \in \mathbb{R}^{[n]\times X}$ as $(\alpha\boldsymbol{x})(i,x) = \alpha\boldsymbol{x}(i,x)$, $(\boldsymbol{x}+\boldsymbol{y})(i,x) = \boldsymbol{x}(i,x) + \boldsymbol{y}(i,x)$, and $\boldsymbol{x}^-(i,x) = \min\{0, \boldsymbol{x}(i,x)\}$ for all $i \in [n]$ and $x \in X$, respectively. If $\boldsymbol{x}(i,x) \leq 0$ for all $i \in [n]$ and $x \in X$ we write $\boldsymbol{x} \leq 0$. For $i \in [n]$ we use $\boldsymbol{x}(i)$ to denote the function $x' \in \mathbb{R}^X$ such that $\boldsymbol{x}(i,x) = x'(x)$ for all $x \in X$.

For $i \in [n]$ and $a \in A$ let $\boldsymbol{\chi_{i,a}} \in \mathbb{R}^{[n]\times A}$ be the vector such that $\boldsymbol{\chi_{i,a}}(i,a) = 1$ and $\boldsymbol{\chi_{i,a}}(i',a') = 0$ for $(i',a') \neq (i,a)$. (So $\boldsymbol{\chi_{i,a}}$ is the unit vector for the coordinate (i,a).) Similarly, we use $\boldsymbol{\chi_i}$ to denote the vector $\sum_{a\in A} \boldsymbol{\chi}_{i,a}$. For a vector $\boldsymbol{x} \in \mathbb{R}^{[n]\times A}$ and tuple $\boldsymbol{y} \in \mathcal{M}^n$ we define

$$\boldsymbol{x}(\boldsymbol{y}) = \sum_{i=1}^{n} g(\boldsymbol{x}(i), \boldsymbol{y}(i))$$

where the function $g : \mathbb{R}^A \times \mathcal{M} \to \mathbb{R}$ is defined by

$$g(x,y) = \begin{cases} 0 & \text{if } y = 0_{\mathcal{M}}, \\ x(y) & \text{if } y \in A, \text{ and} \\ \max\limits_{a,a' \in A, a \neq a'} x(a) + x(a') & \text{otherwise (if } y = 1_{\mathcal{M}}). \end{cases}$$

(This should be compared to how applying a vector to a subset is defined for submodular set functions, see Section 8.2.) For $\boldsymbol{x}, \boldsymbol{x'} \in \mathbb{R}^{[n]\times A}$ we denote the usual scalar product by $\langle \boldsymbol{x}, \boldsymbol{x'} \rangle$, so

$$\langle \boldsymbol{x}, \boldsymbol{x'} \rangle = \sum_{i=1}^{n} \sum_{x \in A} \boldsymbol{x}(i,x)\boldsymbol{x'}(i,x).$$

Let f be a submodular function on $\mathcal{M}^n$ such that $f(\mathbf{0}_{\boldsymbol{\mathcal{M}^n}}) \geq 0$. We define $P_M(f)$ and $B_M(f)$ as follows,

$$P_M(f) = \left\{ \boldsymbol{x} \in \mathbb{R}^{[n]\times A} \;\middle|\; \forall \boldsymbol{y} \in \mathcal{M}^n, \boldsymbol{x}(\boldsymbol{y}) \leq f(\boldsymbol{y}) \right\}, \text{ and}$$
$$B_M(f) = \left\{ \boldsymbol{x} \in \mathbb{R}^{[n]\times A} \;\middle|\; \boldsymbol{x} \in P_M(f), \boldsymbol{x}(\mathbf{1}_{\boldsymbol{\mathcal{M}^n}}) = f(\mathbf{1}_{\boldsymbol{\mathcal{M}^n}}) \right\}.$$

Due to the definition of g it is not hard to see that $P_M(f)$ is a polyhedron. Note that if $\boldsymbol{t}$ contains at least one $1_{\mathcal{M}}$, then $\boldsymbol{t}$ induces more than one linear inequality and if $\boldsymbol{t}$ contains no $1_{\mathcal{M}}$, then $\boldsymbol{t}$ induces exactly one linear inequality. In general, a tuple with m occurrences of $1_{\mathcal{M}}$ induces $\binom{|A|}{2}^m$ linear inequalities. We use $I(\boldsymbol{t})$ to denote the set of all vectors $\boldsymbol{e} \in \mathbb{R}^{[n]\times A}$ such that $\boldsymbol{e}$ represents an inequality induced by $\boldsymbol{t}$ (that is, an inequality of the form $\langle \boldsymbol{e}, \boldsymbol{x} \rangle \leq f(\boldsymbol{t})$, where $\boldsymbol{e} \in I(\boldsymbol{t})$).

We will also need the following definition.

Definition 8.1 (Unified vector for diamonds)**.** *A vector $\boldsymbol{x} \in \mathbb{R}^A$ is* unified *if there is an atom $p \in A$ such that*

- *if $x, y \in A \setminus \{p\}$, then $\boldsymbol{x}(x) = \boldsymbol{x}(y)$; and*
- *if $x \in A$, then $\boldsymbol{x}(p) \geq \boldsymbol{x}(x)$.*

We extend the definition of unified vectors to the vectors in $\mathbb{R}^{[n]\times A}$ by saying that $\boldsymbol{x} \in \mathbb{R}^{[n]\times A}$ is unified if $x \mapsto \boldsymbol{x}(i,x)$ is unified for each $i \in [n]$.

If the submodular inequality is strict for all incomparable pairs of elements then we say that the function is *strictly submodular*.

Given a vector $\boldsymbol{x} \in P_M(f)$ we say that a tuple $\boldsymbol{t} \in \mathcal{M}^n$ such that $\boldsymbol{x}(\boldsymbol{t}) = f(\boldsymbol{t})$ is *$\boldsymbol{x}$-tight*. The following lemma states that if $\boldsymbol{a}$ and $\boldsymbol{b}$ are $\boldsymbol{x}$-tight, then so are $\boldsymbol{a} \sqcap \boldsymbol{b}$ and $\boldsymbol{a} \sqcup \boldsymbol{b}$. This simple result will be used repeatedly in the subsequent parts of this chapter.

Lemma 8.2. *Let $f : \mathcal{M}^n \to \mathbb{R}$ be a submodular function. Let $\boldsymbol{x} \in P_M(f)$ be a vector and let $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{M}^n$ be $\boldsymbol{x}$-tight tuples. Then, $\boldsymbol{a} \sqcup \boldsymbol{b}$ and $\boldsymbol{a} \sqcap \boldsymbol{b}$ are $\boldsymbol{x}$-tight.*

Proof.

$$\boldsymbol{x}(\boldsymbol{a} \sqcup \boldsymbol{b}) + \boldsymbol{x}(\boldsymbol{a} \sqcap \boldsymbol{b}) \leq f(\boldsymbol{a} \sqcup \boldsymbol{b}) + f(\boldsymbol{a} \sqcap \boldsymbol{b}) \leq f(\boldsymbol{a}) + f(\boldsymbol{b}) = \boldsymbol{x}(\boldsymbol{a}) + \boldsymbol{x}(\boldsymbol{b}).$$

The first inequality follows from $\boldsymbol{x} \in P_M(f)$, the second follows from the submodularity of f. The equality follows from the assumptions in the lemma. Note that $\boldsymbol{x}(\boldsymbol{a}) + \boldsymbol{x}(\boldsymbol{b}) \leq \boldsymbol{x}(\boldsymbol{a} \sqcup \boldsymbol{b}) + \boldsymbol{x}(\boldsymbol{a} \sqcap \boldsymbol{b})$. Since $\boldsymbol{x}(\boldsymbol{a} \sqcup \boldsymbol{b}) \leq f(\boldsymbol{a} \sqcup \boldsymbol{b})$ and $\boldsymbol{x}(\boldsymbol{a} \sqcap \boldsymbol{b}) \leq f(\boldsymbol{a} \sqcap \boldsymbol{b})$, it follows that $\boldsymbol{x}(\boldsymbol{a} \sqcup \boldsymbol{b}) = f(\boldsymbol{a} \sqcup \boldsymbol{b})$ and $\boldsymbol{x}(\boldsymbol{a} \sqcap \boldsymbol{b}) = f(\boldsymbol{a} \sqcap \boldsymbol{b})$. □

8.4 A Min–max Theorem

The main results in this section are Theorem 8.5 and Theorem 8.7. We start by two lemmas which shows that $B_M(f)$ is non-empty for any submodular function which maps the bottom of the lattice to a nonnegative value.

Lemma 8.3. *Let $f : \mathcal{M}^n \to \mathbb{R}$ be a submodular function. There is a vector $\boldsymbol{x} \in \mathbb{R}^{[n] \times A}$ such that*

- *$\boldsymbol{x}$ is unified; and*
- *$\boldsymbol{x}(\boldsymbol{v_i}) = f(\boldsymbol{v_i})$ for all $i \in [n]$; and*
- *$\boldsymbol{x}(\boldsymbol{v_{i-1}}[i = p_i]) = f(\boldsymbol{v_{i-1}}[i = p_i])$ for all $i \in [n]$, where for $i \in [n]$, p_i is the atom in Definition 8.1 for the vector $x \mapsto \boldsymbol{x}(i, x)$.*

Furthermore, if f is integer-valued, then $\boldsymbol{x}$ can be chosen to be integer-valued.

Proof. Given a submodular $f : \mathcal{M}^n \to \mathbb{R}$ we will construct a vector $\boldsymbol{x}$ which satisfies the requirements in the lemma. To do this we define a sequence of atoms $p_1, p_2, \ldots, p_n$. For $i \in [n]$, choose p_i so that

$$p_i \in \arg\max_{a \in A} f(\boldsymbol{v_{i-1}}[i = a]).$$

Set $\boldsymbol{x}(1, p_1) = f(\boldsymbol{v_0}[1 = p_1])$ and for $i \in [n], i \neq 1$ set

$$\boldsymbol{x}(i, p_i) = f(\boldsymbol{v_{i-1}}[i = p_i]) - f(\boldsymbol{v_{i-1}}). \tag{8.2}$$

For $i \in [n]$ and $a \in A, a \neq p_i$ set

$$\boldsymbol{x}(i, a) = f(\boldsymbol{v_i}) - f(\boldsymbol{v_{i-1}}[i = p_i]). \tag{8.3}$$

We continue by establishing two claims.

Claim A. For all $i \in [n]$ and $a \in A$ we have $\boldsymbol{x}(i, p_i) \geq \boldsymbol{x}(i, a)$.

Proof. Assume, without loss of generality, that $a \neq p_i$. We now get

$$\begin{aligned} & f(\boldsymbol{v_i}) + f(\boldsymbol{v_{i-1}}) && \leq \\ & f(\boldsymbol{v_{i-1}}[i = p_i]) + f(\boldsymbol{v_{i-1}}[i = a]) && \leq \\ & 2f(\boldsymbol{v_{i-1}}[i = p_i]) \end{aligned}$$

where the first inequality holds due to the submodularity of f and the second inequality follows from our choice of p_i. This is equivalent to

$$\begin{aligned}\boldsymbol{x}(i,a) &= f(\boldsymbol{v_i}) - f(\boldsymbol{v_{i-1}}[i = p_i]) \\ &\leq f(\boldsymbol{v_{i-1}}[i = p_i]) - f(\boldsymbol{v_{i-1}}) \\ &= \boldsymbol{x}(i, p_i)\end{aligned}$$

which is what we wanted to prove.

For $i \in [n]$ and $j \in \{1,2\}$ we define $c_{i,j}$ as $c_{i,1} = p_i$ and $c_{i,2} = 1_{\mathcal{M}}$.

Claim B. For all $i \in [n]$ and $j \in \{1,2\}$ we have $\boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c_{i,j}]) = f(\boldsymbol{v_{i-1}}[i = c_{i,j}])$.

Proof. We prove this by induction over the pairs (i,j) ordered lexicographically (so $(i,j) \leq (i',j')$ if and only if $i < i'$ or $(i = i'$ and $j \leq j')$). With the pair (i,j) we associate the tuple $\boldsymbol{v_{i-1}}[i = c_{i,j}]$. Note that $(i,j) \leq (i',j')$ if and only if $\boldsymbol{v_{i-1}}[i = c_{i,j}] \sqsubseteq \boldsymbol{v_{i'-1}}[i' = c_{i',j'}]$. As $p_1 \in \arg\max_{a \in A} f(\boldsymbol{v_0}[1 = a])$ the claim clearly holds for $(i,j) = (1,1)$. Now assume that it holds for all pairs (i',j') such that $(i',j') \leq (i,j)$. If $j = 1$ then the next pair is $(i,2)$ and we get

$$\begin{aligned}\boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c_{i,2}]) &= \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = p_i]) + \boldsymbol{x}(i,a) \\ &= f(\boldsymbol{v_{i-1}}[i = p_i]) + f(\boldsymbol{v_i}) - f(\boldsymbol{v_{i-1}}[i = p_i]) \\ &= f(\boldsymbol{v_i}).\end{aligned}$$

Here the first inequality follows from the definition of $\boldsymbol{x}(\cdot)$ and Claim A. The second equality follows from the induction hypothesis and (8.3). If $j = 2$ the next pair is $(i+1,1)$ and we get

$$\begin{aligned}\boldsymbol{x}(\boldsymbol{v_i}[i+1 = c_{i+1,1}]) &= \boldsymbol{x}(\boldsymbol{v_i}) + \boldsymbol{x}(i+1, p_{i+1}) \\ &= f(\boldsymbol{v_i}) + f(\boldsymbol{v_i}[i+1 = p_{i+1}]) - f(\boldsymbol{v_i}) \\ &= f(\boldsymbol{v_i}[i+1 = p_{i+1}])\end{aligned}$$

The first equality follows from the definition of $\boldsymbol{x}(\cdot)$. The second equality follows from the induction hypothesis and (8.2).

By Claim A it follows that $\boldsymbol{x}$ is unified. By Claim B $\boldsymbol{x}$ satisfies the second condition in the statement of the lemma. It is easy to see that if f is integer-valued, then so is $\boldsymbol{x}$. □

Lemma 8.4. *Let $f : \mathcal{M}^n \to \mathbb{R}$ be submodular such that $f(\mathbf{0}_{\mathcal{M}^n}) \geq 0$. Let $\boldsymbol{x}$ be a vector in $\mathbb{R}^{[n] \times A}$. If for each $i \in [n]$ there is an atom p_i such that*

- *for all $i \in [n]$, $\boldsymbol{x}(\boldsymbol{v_i}) = f(\boldsymbol{v_i})$; and*
- *for all $i \in [n]$, $\boldsymbol{x}(\boldsymbol{v_{i-1}}[i = p_i]) = f(\boldsymbol{v_{i-1}}[i = p_i])$,*

then $\boldsymbol{x} \in B_M(f)$.

Proof. As $\boldsymbol{x}(\mathbf{1}_{\mathcal{M}^n}) = f(\mathbf{1}_{\mathcal{M}^n})$ it is sufficient to show that $\boldsymbol{x} \in P_M(f)$. For $i \in [n]$ and $j \in \{1,2\}$ we define $c_{i,j}$ as $c_{i,1} = p_i$ and $c_{i,2} = 1_{\mathcal{M}}$. We will prove by induction that $\boldsymbol{x}(\boldsymbol{y}) \leq f(\boldsymbol{y})$ for all $\boldsymbol{y} \in \mathcal{M}^n$. As in the proof of Claim B in Lemma 8.3 the induction will be over the pairs $[n] \times \{1,2\}$ ordered lexicographically. With the pair (i,j) we associate the tuples $\boldsymbol{y}$ such that $\boldsymbol{y} \sqsubseteq \boldsymbol{v_{i-1}}[i = c_{i,j}]$.

As

$$\boldsymbol{x}(\boldsymbol{v_0}) = \boldsymbol{x}(\mathbf{0}_{\mathcal{M}^n}) = 0 \text{ and } f(\mathbf{0}_{\mathcal{M}^n}) \geq 0$$

and

$$\boldsymbol{x}(\boldsymbol{v_0}[1 = p_1]) = f(\boldsymbol{v_0}[1 = p_1])$$

the statement holds for the pair $(1,1)$ (which corresponds to $\boldsymbol{y} \sqsubseteq \mathbf{0}_{\mathcal{M}^n}[1 = p_1]$). Let $i \in [n]$, $j \in \{1,2\}$, and $\boldsymbol{y} \in \mathcal{M}^n, \boldsymbol{y} \sqsubseteq \boldsymbol{v_{i-1}}[i = c_{i,j}]$ and assume that the inequality holds for all $\boldsymbol{y'} \in \mathcal{M}^n$ such that $\boldsymbol{y'} \sqsubseteq \boldsymbol{v_{i'-1}}[i' = c_{i',j'}]$ where (i',j') is the predecessor to the pair (i,j). We will prove that the inequality holds for all $\boldsymbol{y} \sqsubseteq \boldsymbol{v_{i-1}}[i = c_{i,j}]$.

To simplify the notation a bit we let $y = \boldsymbol{y}(i)$. If $y = 0_{\mathcal{M}}$ we are already done, so assume that $y \neq 0_{\mathcal{M}}$. If $y = p_i$ let $c = 0_{\mathcal{M}}$, if $y \in A, y \neq p_i$ let $c = p_i$ and otherwise, if $y = 1_{\mathcal{M}}$ let $c = p_i$. Now,

$$\begin{aligned}
\boldsymbol{x}(\boldsymbol{y}) &\leq \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = y \sqcup c]) - \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c]) + \boldsymbol{x}(\boldsymbol{y}[i = y \sqcap c]) \\
&\leq \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = y \sqcup c]) - \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c]) + f(\boldsymbol{y}[i = y \sqcap c]) \\
&\leq f(\boldsymbol{v_{i-1}}[i = y \sqcup c]) - f(\boldsymbol{v_{i-1}}[i = c]) + f(\boldsymbol{y}[i = y \sqcap c]) \\
&\leq f(\boldsymbol{y}).
\end{aligned}$$

The first inequality follows from the supermodularity of $\boldsymbol{x}$. The second inequality follows from the induction hypothesis and the fact that $y \sqcap c \sqsubset y$ and $y \sqcap c \in \{0_{\mathcal{M}}, p_i\}$. The third inequality follows from $y \sqcup c, c \in \{0_{\mathcal{M}}, p_i, 1_{\mathcal{M}}\}$ and the assumptions in the statement of the lemma. Finally, the last inequality follows from the submodularity of f. □

In the proof of Lemma 8.3 the vector $\boldsymbol{x} \in \mathbb{R}^{[n]\times A}$ is constructed with a greedy approach—we order the coordinates of the vector, $[n] \times A$, in a certain way and then set each component to its maximum value subject to the constraints given in the definition of $B_M(f)$. We remark that the greedy algorithm *does not* solve the optimisation problem for $P_M(f)$. As an example, let $\mathcal{M}_3 = (\{0_{\mathcal{M}}, 1_{\mathcal{M}}, a, b, c\}, \sqcap, \sqcup)$ be a diamond and let $f : \mathcal{M}_3 \to \mathbb{R}$ be defined as $f(0_{\mathcal{M}}) = 0$, $f(a) = f(b) = f(c) = f(1_{\mathcal{M}}) = 1$. The function f is submodular. Now let $\boldsymbol{c} \in \mathbb{R}^{[1]\times A}$ and $\boldsymbol{c}(1,a) = \boldsymbol{c}(1,b) = \boldsymbol{c}(1,c) = 1$. From the greedy algorithm we will get a vector $\boldsymbol{x} \in \mathbb{R}^{[1]\times A}$ such that $\boldsymbol{x}(1,a) = 1$ and $\boldsymbol{x}(1,b) = \boldsymbol{x}(1,c) = 0$ (or some permutation of this vector). However, the solution to $\max\langle \boldsymbol{c}, \boldsymbol{y}\rangle, \boldsymbol{y} \in P_M(f)$ is $\boldsymbol{y}(1,a) = \boldsymbol{y}(1,b) = \boldsymbol{y}(1,c) = 1/2$ and $3/2 = \langle \boldsymbol{c}, \boldsymbol{y}\rangle > \langle \boldsymbol{c}, \boldsymbol{x}\rangle = 1$. This example also shows that the vertices of $P_M(f)$ are not necessarily integer-valued. This should be compared to submodular set functions, where the corresponding optimisation problem *is* solved by the greedy algorithm. [116]

Given an algorithm which solves the optimisation problem over $P_M(f)$ in time polynomial in n we can use the equivalence of optimisation and separation given by the Ellipsoid algorithm to solve the separation problem for $P_M(f)$ in polynomial time. With such an algorithm we can decide if $\mathbf{0} \in P_M(f)$ or not and by a binary search we can find a minimiser of f in polynomial time. So a polynomial time algorithm for the optimisation problem over $P_M(f)$ would be desirable. (The approach outlined above can be used to minimise submodular set functions, see [72] or, e.g., [73].) We present a pseudopolynomial-time algorithm for the optimisation problem in Section 8.7 which uses this technique.

We are now ready to state the two main theorems of this section.

Theorem 8.5. *Let $f : \mathcal{M}^n \to \mathbb{R}$ be a submodular function such that $f(\mathbf{0}_{\mathcal{M}^n}) = 0$, then*

$$\min_{\boldsymbol{x} \in \mathcal{M}^n} f(\boldsymbol{x}) = \max \left\{ \boldsymbol{z}(\mathbf{1}_{\mathcal{M}^n}) \;\middle|\; \boldsymbol{z} \in P_M(f), \boldsymbol{z} \leq 0,\ \boldsymbol{z} \text{ is unified} \right\}.$$

More over, if f is integer-valued then there is an integer-valued vector $\boldsymbol{z}$ which maximises the right hand side.

Proof. If $\boldsymbol{z} \in P_M(f)$, $\boldsymbol{z} \leq 0$, and $\boldsymbol{z}$ is unified, then

$$\boldsymbol{z}(\mathbf{1}_{\mathcal{M}^n}) \leq \boldsymbol{z}(\boldsymbol{y}) \leq f(\boldsymbol{y})$$

for any $\boldsymbol{y} \in \mathcal{M}^n$. Hence, LHS $\geq$ RHS holds. Consider the function $f' : \mathcal{M}^n \to \mathbb{R}$ defined by

$$f'(\boldsymbol{x}) = \min_{\boldsymbol{y} \sqsubseteq \boldsymbol{x}} f(\boldsymbol{y}).$$

Then $P_M(f') \subseteq P_M(f)$.

Claim A. f' is submodular.

Proof. Let $\boldsymbol{x'}, \boldsymbol{y'} \in \mathcal{M}^n$ and let $\boldsymbol{x} \sqsubseteq \boldsymbol{x'}, \boldsymbol{y} \sqsubseteq \boldsymbol{y'}$ be tuples such that $f'(\boldsymbol{x'}) = f(\boldsymbol{x})$ and $f'(\boldsymbol{y'}) = f(\boldsymbol{y})$. Now,

$$\begin{aligned} f'(\boldsymbol{x'}) + f'(\boldsymbol{y'}) = f(\boldsymbol{x}) + f(\boldsymbol{y}) &\geq f(\boldsymbol{x} \sqcap \boldsymbol{y}) + f(\boldsymbol{x} \sqcup \boldsymbol{y}) \\ &\geq f'(\boldsymbol{x'} \sqcap \boldsymbol{y'}) + f'(\boldsymbol{x'} \sqcup \boldsymbol{y'}) \end{aligned}$$

where the first equality follows from the definition of f', $\boldsymbol{x}$ and $\boldsymbol{y}$, the first inequality follows from the submodularity of f and the second inequality from the definition of f' and $\boldsymbol{x} \sqcap \boldsymbol{y} \sqsubseteq \boldsymbol{x'} \sqcap \boldsymbol{y'}$ and $\boldsymbol{x} \sqcup \boldsymbol{y} \sqsubseteq \boldsymbol{x'} \sqcup \boldsymbol{y'}$.

Claim B. For any $\boldsymbol{z} \in P_M(f')$ we have $\boldsymbol{z} \leq 0$.

Proof. As $f(\mathbf{0}_{\mathcal{M}^n}) = 0$ we have $f'(\boldsymbol{x}) \leq 0$ for any $\boldsymbol{x} \in \mathcal{M}^n$. For $i \in [n]$ and $a \in A$ define $\boldsymbol{t_{i,a}} \in \mathcal{M}^n$ such that $\boldsymbol{t_{i,a}}(j) = 0_{\mathcal{M}}$ for $j \in [n], j \neq i$ and $\boldsymbol{t_{i,a}}(i) = a$. It follows from $\boldsymbol{z} \in P_M(f')$ that we have $\boldsymbol{z}(\boldsymbol{t_{i,a}}) = \boldsymbol{z}(i, a) \leq f'(\boldsymbol{t_{i,a}}) \leq 0$ for any $a \in A$ and $i \in [n]$.

Claim C. Any $\boldsymbol{z} \in B_M(f') \subseteq P_M(f')$ satisfies $\boldsymbol{z}(\mathbf{1}_{\mathcal{M}^n}) = f'(\mathbf{1}_{\mathcal{M}^n})$.

Proof. Follows from the definition of $B_M(f')$.

From Lemma 8.3 and Lemma 8.4 it now follows that $B_M(f')$ is non-empty, indeed there is a unified vector $\boldsymbol{z} \in B_M(f')$. As $B_M(f') \subseteq P_M(f') \subseteq P_M(f)$ it follows that $\boldsymbol{z} \in P_M(f)$. Claim B implies that $\boldsymbol{z} \leq 0$ and from Claim C we get $\boldsymbol{z}(\mathbf{1}_{\mathcal{M}^n}) = f'(\mathbf{1}_{\mathcal{M}^n})$. LHS $\leq$ RHS now follows from $f'(\mathbf{1}_{\mathcal{M}^n}) = \min_{\boldsymbol{x} \in \mathcal{M}^n} f(\boldsymbol{x})$, which in turn follows from the definition of f'.

To prove the existence of an integer-valued vector, note that the vector from Lemma 8.3 is integer-valued if f' is integer-valued and f' is integer-valued if f is integer-valued. □

We can reformulate Theorem 8.5 to relate the minimum of a submodular function f to the maximum of a certain function defined over the polyhedron $\{\boldsymbol{x} \in P_M(f) \mid \boldsymbol{x} \leq 0\}$. To do this we define a function $S : \mathbb{R}^{[n] \times A} \to \mathbb{R}$ as follows

$$S(\boldsymbol{x}) = \sum_{i=1}^{n} \min_{a \in A} \boldsymbol{x}(i, a) + \max_{a \in A} \boldsymbol{x}(i, a).$$

We then get the following corollary.

Corollary 8.6.

$$\min_{\boldsymbol{y} \in \mathcal{M}^n} f(\boldsymbol{y}) = \max \left\{ S(\boldsymbol{z}) \;\middle|\; \boldsymbol{z} \in P_M(f), \boldsymbol{z} \leq 0 \right\}.$$

Proof. Follows from Theorem 8.5 by two observations. If $\boldsymbol{z}$ is unified, then $\boldsymbol{z}(\mathbf{1}_{\mathcal{M}^n}) = S(\boldsymbol{z})$. Furthermore, any vector $\boldsymbol{z}$ can be turned into a unified vector $\boldsymbol{z'}$ such that $\boldsymbol{z'} \leq \boldsymbol{z}$ and $S(\boldsymbol{z}) = \boldsymbol{z'}(\mathbf{1}_{\mathcal{M}^n})$. To construct $\boldsymbol{z'}$ from $\boldsymbol{z}$, for each $i \in [n]$, choose some $p_i \in \arg\max_{a \in A} \boldsymbol{z}(i, a)$ and let $\boldsymbol{z'}(i, p_i) = \boldsymbol{z}(i, p_i)$ and for $a \in A, a \neq p_i$ let $\boldsymbol{z'}(i, a) = \min_{a \in A} \boldsymbol{z}(i, a)$. □

One might ask if there is any reason to believe that the min–max characterisation given by Theorem 8.5 is the "right" way to look at this problem. That is, can this min–max relation give insight into the complexity of minimising submodular functions over diamonds? Theorem 8.5 is used in Section 8.6 to get a good characterisation of submodular function minimisation over diamonds, so it certainly gets us somewhere. In Section 8.7 we present a pseudopolynomial-time algorithm which uses $P_M(f)$, but it does not use Theorem 8.5. Additionally, Theorem 8.5 is in some sense fairly similar to (8.1). In particular, in both cases the vectors are functions from the atoms of the lattices to the real numbers and when a vector is applied to a tuple (or a subset) it is computed as a sum over the coordinates of the vector and the tuple. Furthermore, in this sum the bottom of the lattice ($0_{\mathcal{M}}$ in the diamond case and an omitted element in the set case) do not contribute to the sum. There are of course differences as well. The most obvious one is, perhaps, that there is no element in the set case analogous to $1_{\mathcal{M}}$ in the diamond case. As far as we know, all combinatorial algorithms for submodular set function minimisation is based on (8.1). Considering the similarity between Theorem 8.5 and (8.1) one could hope that Theorem 8.5 could be the basis for a polynomial time combinatorial algorithm for SFM($\mathcal{M}$).

The following theorem is an analogue to the second equality in Edmonds' min–max theorem for submodular set functions (8.1).

Theorem 8.7. *Let $f : \mathcal{M}^n \to \mathbb{R}$ be a submodular function such that $f(\mathbf{0}_{\mathcal{M}^n}) = 0$, then*

$$\begin{aligned}\min_{\boldsymbol{x}\in\mathcal{M}^n} f(\boldsymbol{x}) &= \max\left\{\boldsymbol{z}(\mathbf{1}_{\mathcal{M}^n}) \;\middle|\; \boldsymbol{z} \in P_M(f), \boldsymbol{z} \le 0,\ \boldsymbol{z} \text{ is unified}\right\} \\ &= \max\left\{\boldsymbol{x}^-(\mathbf{1}_{\mathcal{M}^n}) \;\middle|\; \boldsymbol{x} \in B_M(f),\ \boldsymbol{x}^- \text{ is unified}\right\}.\end{aligned}$$

Proof. We prove that

$$\begin{aligned}&\max\left\{\boldsymbol{z}(\mathbf{1}_{\mathcal{M}^n}) \;\middle|\; \boldsymbol{z} \in P_M(f), \boldsymbol{z} \le 0, \boldsymbol{z} \text{ is unified}\right\} = \\ &\max\left\{\boldsymbol{x}^-(\mathbf{1}_{\mathcal{M}^n}) \;\middle|\; \boldsymbol{x} \in B_M(f), \boldsymbol{x} \text{ is unified}\right\}.\end{aligned}$$

The result then follows from Theorem 8.5.

Let $\boldsymbol{x}$ be a vector which maximises the right hand side. It is clear that $\boldsymbol{x}^- \in P_M(f)$, $\boldsymbol{x}^- \le 0$, and that $\boldsymbol{x}^-$ is unified. It follows that LHS $\ge$ RHS.

Conversely, let $\boldsymbol{z}$ be a vector which maximises the left hand side. We will define a sequence of vectors $\boldsymbol{x_0}, \boldsymbol{x_1}, \ldots$. We start with $\boldsymbol{x_0} = \boldsymbol{z}$ and for $j \ge 0$ we define $\boldsymbol{x_{j+1}}$ from $\boldsymbol{x_j}$ according to the construction below.

1. If there is some $i \in [n]$ and $p \in \arg\max_{a\in A} \boldsymbol{x}(i,a)$ such that $\alpha' > 0$ where
$$\alpha' = \max\{\alpha \in \mathbb{R} \mid \boldsymbol{x} + \alpha\boldsymbol{\chi_{i,p}} \in P_M(f)\},$$
then let $\boldsymbol{x_{j+1}} = \boldsymbol{x_j} + \alpha' \cdot \boldsymbol{\chi_{i,p}}$.

2. Otherwise, if there is some $i \in [n]$ and $p \in \arg\max_{a\in A} \boldsymbol{x}(i,a)$ such that $\alpha' > 0$ where
$$\alpha' = \max\{\alpha \in \mathbb{R} \mid \boldsymbol{x_j} + \alpha \cdot (\boldsymbol{\chi_i} - \boldsymbol{\chi_{i,p}}) \in P_M(f)\},$$
then let a be some atom distinct from p, let $m = \min\{\alpha', \boldsymbol{x}(i,p) - \boldsymbol{x}(i,a)\}$, and let $\boldsymbol{x_{j+1}} = \boldsymbol{x_j} + m \cdot (\boldsymbol{\chi_i} - \boldsymbol{\chi_{i,p}})$.

We make four observations of this construction.

- If we reach the second step, then $\arg\max_{a\in A} \boldsymbol{x}(i,a)$ is a one element set.
- For every j the vector $\boldsymbol{x_j}$ is unified.
- For every j, $\boldsymbol{x_{j+1}} \ge \boldsymbol{x_j}$.
- For every j, $\boldsymbol{x_j} \in P(f)$.

These observations all follow directly from the construction above. It is not hard to convince oneself that there is an integer m such that $\boldsymbol{x_m} = \boldsymbol{x_{m+1}}$ (and thus all vectors constructed after m are equal). To see this, note that for a fixed $i \in [n]$ if some atom a is increased in step 1, then this atom will never be increased again at coordinate i (neither in step 1 nor in step 2). Furthermore, step 2 will be applicable at most once for each $i \in [n]$.

Let $\boldsymbol{y}$ denote the vector $\boldsymbol{x_m}$. Note that $\boldsymbol{x_{j+1}}^-(\mathbf{1}_{\boldsymbol{\mathcal{M}^n}}) \geq \boldsymbol{x_j}^-(\mathbf{1}_{\boldsymbol{\mathcal{M}^n}})$ for all j. Hence in particular $\boldsymbol{y}^-(\mathbf{1}_{\boldsymbol{\mathcal{M}^n}}) \geq \boldsymbol{z}(\mathbf{1}_{\boldsymbol{\mathcal{M}^n}})$. As we have already proved that LHS $\geq$ RHS it now remains to prove that $\boldsymbol{y} \in B_M(f)$. As we already know that $\boldsymbol{y} \in P_M(f)$ this reduces to proving $\boldsymbol{y}(\mathbf{1}_{\boldsymbol{\mathcal{M}^n}}) = f(\mathbf{1}_{\boldsymbol{\mathcal{M}^n}})$.

Let $\boldsymbol{p}$ be a tuple such that for $i \in [n]$ we have $\boldsymbol{p}(i) \in \arg\max_{a \in A} \boldsymbol{y}(i,a)$. As $\boldsymbol{y} = \boldsymbol{x_m} = \boldsymbol{x_{m+1}}$, it follows that for each $k \in [n]$ there is an atom $a \in A, a \neq \boldsymbol{p}(k)$ and tuples $\boldsymbol{t_k}, \boldsymbol{t'_k} \in \mathcal{M}^n, \boldsymbol{p}(k) \sqsubseteq \boldsymbol{t_k}(k), a \sqsubseteq \boldsymbol{t'_k}(k)$ such that $\boldsymbol{t_k}$ and $\boldsymbol{t'_k}$ are $\boldsymbol{y}$-tight. Now let,

$$\boldsymbol{t} = \bigsqcup_{k \in [n]} \boldsymbol{t_k} \sqcup \boldsymbol{t'_k}.$$

As $\boldsymbol{y} \in P_M(f)$ it follows from Lemma 8.2 that $\boldsymbol{y}(\boldsymbol{t}) = f(\boldsymbol{t})$. Note that for each $k \in [n]$ we have $(\boldsymbol{t_k} \sqcup \boldsymbol{t'_k})(k) = 1_{\mathcal{M}}$, it follows that $\boldsymbol{t} = \mathbf{1}_{\boldsymbol{\mathcal{M}^n}}$ and hence $\boldsymbol{y} \in B_M(f)$. We conclude that LHS $\leq$ RHS. □

8.5 The Ellipsoid Algorithm

In this section we present some definitions and results which are related to the Ellipsoid algorithm. We will use these results in our algorithms for submodular function minimisation, both over diamonds in this chapter and over general modular lattices in Chapter 9. All results presented here are from the book [73]. As in [73] we make the general assumption that for any oracle O there is an integer c such that when O is given input data of length n the length of the output is $O(n^c)$.

We need a few basic facts about polyhedrons. Let $P \subseteq \mathbb{R}^n$ be a polyhedron. The *lineality space* of P, denoted by lin.space P, is the set of vectors $\boldsymbol{x}$ such that there is a vector $\boldsymbol{y} \in P$ and $\lambda\boldsymbol{x} + \boldsymbol{y} \in P$ for all $\lambda \in \mathbb{R}$. The *characteristic cone* of P, denoted by char.cone P, is the set of vectors $\boldsymbol{x} \in \mathbb{R}^n$ such that for all $\boldsymbol{y} \in P$ and $\lambda \geq 0$ we have $\lambda\boldsymbol{x} + \boldsymbol{y} \in P$. Given a set of vectors $X \subseteq \mathbb{R}^n$ we use conv(X) to denote the convex hull of X and cone(X) to denote

$$\{\lambda_1 x_1 + \ldots + \lambda_t x_t \mid t \in \mathbb{N}, x_1, \ldots, x_t \in X, \lambda_1, \ldots, \lambda_t \geq 0\}.$$

Definition 8.8 (Oracle-polynomial time)**.** *An algorithm A, with access to an oracle O, runs in* oracle-polynomial time *if there is an integer c such that given any input of length n A makes $O(n^c)$ calls to O and performs $O(n^c)$ additional primitive operations.*

Definition 8.9 (Facet- and vertex-complexity [73, Definition 6.2.2a,b])**.** *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and let ϕ and ν be positive integers.*

- *P has* facet-complexity at most *ϕ if there exists a system of linear inequalities with rational coefficients that has solution set P and such that any equation can be encoded with at most ϕ bits. If $P = \mathbb{R}^n$ we require that $\phi \geq n + 1$.*

- *P has* vertex-complexity at most *ν if there exist finite sets V and E of rational vectors such that $P = conv(V) + cone(E)$ and such that each vector in V and E can be encoded with at most ν bits. If $P = \emptyset$ we require that $\nu \geq n$.*

Definition 8.10 (Well-described polyhedron [73, Definition 6.2.2c])**.** *A* well-described polyhedron *is a triple $(P; n, \phi)$ where $P \subseteq \mathbb{R}^n$ is a polyhedron with facet-complexity at most ϕ. The encoding length of $(P; n, \phi)$ is $\phi + n$.*

Definition 8.11 (Strong optimisation problem [73, Definition 6.2.1])**.** *Given a polyhedron $P \subseteq \mathbb{R}^n$ and a vector $\boldsymbol{c} \in \mathbb{Q}^n$, either*

- *assert that P is empty, or*
- *find a vector $\boldsymbol{y} \in P$ maximising $\langle \boldsymbol{c}, \boldsymbol{x} \rangle$ over P, or*
- *find a vector $\boldsymbol{z} \in$ char.cone P such that $\langle \boldsymbol{c}, \boldsymbol{z} \rangle \geq 1$.*

Definition 8.12 (Strong separation problem [73, Definition 2.1.4])**.** *Let $P \subseteq \mathbb{R}^n$ be a polyhedron. Given a vector $\boldsymbol{y} \in \mathbb{R}^n$, decide whether $\boldsymbol{y} \in P$, and if not, find a hyperplane that separates $\boldsymbol{y}$ from P; more exactly, find a vector $\boldsymbol{c} \in \mathbb{R}^n$ such that $\langle \boldsymbol{c}, \boldsymbol{y} \rangle > \max\{\langle \boldsymbol{c}, \boldsymbol{x} \rangle \mid \boldsymbol{x} \in P\}$.*

The following result is a part of Theorem 6.4.9 in [73].

Theorem 8.13. *Let $(P; n, \phi)$ be a well-described polyhedron. The strong separation problem and strong optimisation problem for $(P; n, \phi)$ can be solved in oracle-polynomial time in $n + \phi$ given n, ϕ, and an oracle for the other problem.*

Theorem 8.13 is in some sense quite remarkable: given a description of a polyhedron by only a separation algorithm, it is possible to maximise a linear objective function over the polyhedron in polynomial time. Note that there is no restriction on the number of inequalities defining the polyhedron, there may be an exponential number inequalities and it is still be possible to find the optimum in polynomial time!

The algorithms in Theorem 8.13 require ϕ as a part of the input. When the polyhedron is of the form $P_M(f)$ for some submodular f this amounts to establishing an upper bound on $\max(|f|)$. We will now show how one can obtain such an upper bound.

Let $f : \mathcal{M}^n \to \mathbb{Z}$ be submodular. Let

$$d' = \max\{f(\mathbf{1}_{\boldsymbol{\mathcal{M}^n}}[i = x]) - f(\mathbf{1}_{\boldsymbol{\mathcal{M}^n}}[i = y]) \mid i \in [n], x, y \in \mathcal{M}, x \prec y\}$$

and let $d = \max\{d', 0\}$. Note that we can compute d in time polynomial in n. For all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}^n$ with $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$ we claim that

$$f(\boldsymbol{x}) + d \cdot \rho(\boldsymbol{x}) \leq f(\boldsymbol{y}) + d \cdot \rho(\boldsymbol{y}). \tag{8.4}$$

(The technique we use to prove this claim comes from [133].) We prove that $\boldsymbol{x} \prec \boldsymbol{y}$ implies $f(\boldsymbol{x}) \leq f(\boldsymbol{y}) + d$, this is sufficient to establish (8.4). As $\boldsymbol{x} \prec \boldsymbol{y}$ we have $\boldsymbol{x} = \boldsymbol{y} \sqcap \mathbf{1}_{\mathcal{L}^n}[i = \boldsymbol{x}(i)]$ for some $i \in [n]$. We now get

$$\begin{aligned} f(\boldsymbol{x}) &= f(\boldsymbol{y} \sqcap \mathbf{1}_{\mathcal{M}^n}[i = \boldsymbol{x}(i)]) \\ &\leq f(\boldsymbol{y}) + f(\mathbf{1}_{\mathcal{M}^n}[i = \boldsymbol{x}(i)]) - f(\boldsymbol{y} \sqcup \mathbf{1}_{\mathcal{M}^n}[i = \boldsymbol{x}(i)]) \\ &= f(\boldsymbol{y}) + f(\mathbf{1}_{\mathcal{M}^n}[i = \boldsymbol{x}(i)]) - f(\mathbf{1}_{\mathcal{M}^n}[i = \boldsymbol{y}(i)]) \\ &\leq f(\boldsymbol{y}) + d. \end{aligned}$$

Here the first inequality follows from the submodularity of f and the second equality follows from the fact that $\boldsymbol{x}(i) \sqsubseteq \boldsymbol{y}(i)$. The last inequality follows from our choice of d. Note that d is not too large compared to $\max(|f|)$, indeed $d \leq 2\max(|f|)$. By letting $\boldsymbol{x} = \mathbf{0}_{\mathcal{M}^n}$ in (8.4) we get the bound $f(\mathbf{0}_{\mathcal{M}^n}) \leq \min_{\boldsymbol{z} \in \mathcal{M}^n} f(\boldsymbol{z}) + d \cdot 2n$ and by $\boldsymbol{y} = \mathbf{1}_{\mathcal{M}^n}$ we get $\max_{\boldsymbol{z} \in \mathcal{M}^n} f(\boldsymbol{z}) \leq f(\mathbf{1}_{\mathcal{M}^n}) + d \cdot 2n$. Hence,

$$\max(|f|) \leq \max\{|f(\mathbf{0}_{\mathcal{M}^n}) - d \cdot 2n|, |f(\mathbf{1}_{\mathcal{M}^n}) + d \cdot 2n|\}. \tag{8.5}$$

Note that the RHS of (8.5) is polynomial in n and $\max(|f|)$. We conclude that the running times of the algorithms in Theorem 8.13 are polynomial in $n + \log \max(|f|)$ when given the logarithm of the RHS of (8.5) as ϕ.

8.6 A Good Characterisation

In this section we show that there are membership proofs for $P_M(f)$ which can be checked in time polynomial in n. By using Theorem 8.5 this will lead to the existence of proofs that can be checked in time polynomial in n of the fact that a certain tuple minimises a submodular function.

The following lemma is an important part of the main result in this section.

Lemma 8.14. *Let $f : \mathcal{M}^n \to \mathbb{R}$ be a submodular function and let $\boldsymbol{x}$ be a vertex of $P_M(f)$. Furthermore, assume that $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{M}^n, \boldsymbol{a} \sqsubseteq \boldsymbol{b}$ are $\boldsymbol{x}$-tight and for all $\boldsymbol{t} \in \mathcal{M}^n$ such that $\boldsymbol{a} \sqsubset \boldsymbol{t} \sqsubset \boldsymbol{b}$ the tuple $\boldsymbol{t}$ is not $\boldsymbol{x}$-tight. Then, there is at most one coordinate $i \in [n]$ such that $\boldsymbol{a}(i) = 0_{\mathcal{M}}$ and $\boldsymbol{b}(i) = 1_{\mathcal{M}}$.*

Proof. As $\boldsymbol{x}$ is a vertex of $P_M(f)$ there is some $\boldsymbol{c} \in \mathbb{R}^{[n] \times A}$ such that $\boldsymbol{x}$ is the unique optimum to $\max\langle \boldsymbol{c}, \boldsymbol{y} \rangle, \boldsymbol{y} \in P_M(f)$. Assume, for the sake of contradiction, that there are two coordinates $i, j \in [n], i \neq j$ such that $\boldsymbol{a}(i) = \boldsymbol{a}(j) = 0_{\mathcal{M}}$ and $\boldsymbol{b}(i) = \boldsymbol{b}(j) = 1_{\mathcal{M}}$. We can assume, without loss of generality, that

$$\sum_{x \in A} \boldsymbol{c}(i, x) \geq \sum_{x \in A} \boldsymbol{c}(j, x).$$

Let $\delta > 0$ and let $\boldsymbol{x'} = \boldsymbol{x} + \delta \boldsymbol{\chi_i} - \delta \boldsymbol{\chi_j}$. We cannot have $\boldsymbol{x'} \in P_M(f)$ for any $\delta > 0$, because then either $\boldsymbol{x}$ is not the unique optimum or $\boldsymbol{x}$ is not optimal.

As $\boldsymbol{x'} \notin P_M(f)$ there is some $\boldsymbol{x}$-tight tuple $\boldsymbol{t} \in \mathcal{M}^n$ such that ($\boldsymbol{t}(i) \in A$ and $\boldsymbol{t}(j) = 0_\mathcal{M}$) or ($\boldsymbol{t}(i) = 1_\mathcal{M}$ and $\boldsymbol{t}(j) \in \{0_\mathcal{M}\} \cup A$). In either case, it follows from Lemma 8.2 that $\boldsymbol{t'} = (\boldsymbol{b} \sqcap \boldsymbol{t}) \sqcup \boldsymbol{a}$ is $\boldsymbol{x}$-tight, which is a contradiction as $\boldsymbol{a} \sqsubset \boldsymbol{t'} \sqsubset \boldsymbol{b}$. □

The key lemma of this section is the following result. We will use this lemma together with Lemma 8.14 in the proof of the main result of this section (Theorem 8.19).

Lemma 8.15. *Let n be a positive integer and let $f : \mathcal{M}^n \to \mathbb{R}$ be submodular which is provided to us by a value-giving oracle. Let $\boldsymbol{x} \in \mathbb{R}^{[n]\times A}$ and $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{M}^n$ such that $\boldsymbol{a} \sqsubseteq \boldsymbol{b}$, $\boldsymbol{a}$ is $\boldsymbol{x}$-tight, and there are at most k coordinates $i \in [n]$ such that $\boldsymbol{a}(i) = 0_\mathcal{M}$ and $\boldsymbol{b}(i) = 1_\mathcal{M}$. Under the assumption that for all $\boldsymbol{t} \sqsubseteq \boldsymbol{a}$ we have $\boldsymbol{x}(\boldsymbol{t}) \leq f(\boldsymbol{t})$ it is possible to check if the statement $\forall \boldsymbol{y} \in \mathcal{M}^n : \boldsymbol{y} \sqsubseteq \boldsymbol{b} \Rightarrow \boldsymbol{x}(\boldsymbol{y}) \leq f(\boldsymbol{y})$ is true in time $O\left(|\mathcal{M}|^k \cdot n^c\right)$, for some fixed constant c.*

Proof. We will first show that we can check the inequality for all $\boldsymbol{y}$ which satisfies $\boldsymbol{a} \sqsubseteq \boldsymbol{y} \sqsubseteq \boldsymbol{b}$. We will then prove that if the inequality holds for these $\boldsymbol{y}$, then it holds for all tuples $\sqsubseteq \boldsymbol{b}$.

Let $I \subseteq [n]$ be the set of coordinates such that $\boldsymbol{a}(i) = 0_\mathcal{M}$ and $\boldsymbol{b}(i) = 1_\mathcal{M}$ and let $J = \{j \in [n] \mid \boldsymbol{a}(j) \neq \boldsymbol{b}(j), j \notin I\}$. Let $Z = \{\boldsymbol{z} \in \mathcal{M}^n \mid \forall i \notin I : \boldsymbol{z}(i) = 0_\mathcal{M}\}$. For a subset $Y = \{y_1, y_2, \ldots, y_m\}$ of J and $\boldsymbol{z} \in Z$ define $g_{\boldsymbol{z}} : 2^J \to \mathbb{R}$ as

$$g_{\boldsymbol{z}}(Y) = f(\boldsymbol{a}[y_1 = \boldsymbol{b}(y_1), \ldots, y_m = \boldsymbol{b}(y_m)] \sqcup \boldsymbol{z}).$$

We claim that $g_{\boldsymbol{z}}$ is a submodular set function. Let $\boldsymbol{z} \in Z$ and let $C = \{c_1, c_2, \ldots, c_k\}$ and $D = \{d_1, d_2, \ldots, d_l\}$ be two arbitrary subsets of J. Define $\boldsymbol{c}, \boldsymbol{d} \in \mathcal{M}^n$ as $\boldsymbol{a}[c_1 = \boldsymbol{b}(c_1), \ldots, c_k = \boldsymbol{b}(c_k)] \sqcup \boldsymbol{z}$ and $\boldsymbol{a}[d_1 = \boldsymbol{b}(d_1), \ldots, d_l = \boldsymbol{b}(d_l)] \sqcup \boldsymbol{z}$, respectively. We now get

$$\begin{aligned} g_{\boldsymbol{z}}(C) + g_{\boldsymbol{z}}(D) &= f(\boldsymbol{c}) + f(\boldsymbol{d}) \geq f(\boldsymbol{c} \sqcap \boldsymbol{d}) + f(\boldsymbol{c} \sqcup \boldsymbol{d}) \\ &= g_{\boldsymbol{z}}(C \cap D) + g_{\boldsymbol{z}}(C \cup D). \end{aligned}$$

Hence $g_{\boldsymbol{z}}$ is submodular for each $\boldsymbol{z} \in Z$. For a subset $Y = \{y_1, y_2, \ldots, y_m\}$ of J define $h_{\boldsymbol{z}} : 2^J \to \mathbb{R}$ as

$$h_{\boldsymbol{z}}(Y) = \boldsymbol{x}(\boldsymbol{a}[y_1 = \boldsymbol{b}(y_1), \ldots, y_m = \boldsymbol{b}(y_m)] \sqcup \boldsymbol{z}).$$

We claim that $-h_{\boldsymbol{z}}$ is a submodular set function for each $\boldsymbol{z} \in Z$. As above, let $C = \{c_1, c_2, \ldots, c_k\}$ and $D = \{d_1, d_2, \ldots, d_l\}$ be two arbitrary subsets of J and let $\boldsymbol{c} = \boldsymbol{a}[c_1 = \boldsymbol{b}(c_1), \ldots, c_k = \boldsymbol{b}(c_k)] \sqcup \boldsymbol{z}$ and $\boldsymbol{d} = \boldsymbol{a}[d_1 = \boldsymbol{b}(d_1), \ldots, d_l = \boldsymbol{b}(d_l)] \sqcup \boldsymbol{z}$, then

$$\begin{aligned} h_{\boldsymbol{z}}(C) + h_{\boldsymbol{z}}(D) &= \boldsymbol{x}(\boldsymbol{c}) + \boldsymbol{x}(\boldsymbol{d}) \leq \boldsymbol{x}(\boldsymbol{c} \sqcap \boldsymbol{d}) + \boldsymbol{x}(\boldsymbol{c} \sqcup \boldsymbol{d}) \\ &= h_{\boldsymbol{z}}(C \cap D) + h_{\boldsymbol{z}}(C \cup D). \end{aligned}$$

Hence, $-h_{\boldsymbol{z}}$ is submodular. Let $Y = \{y_1, y_2, \ldots, y_m\}$ be an arbitrary subset of J and let $\boldsymbol{z} \in Z$. For a fixed $\boldsymbol{z} \in Z$ the inequalities

$$\begin{aligned} &\boldsymbol{x}(\boldsymbol{a}[y_1 = \boldsymbol{b}(y_1), \ldots, y_m = \boldsymbol{b}(y_m)] \sqcup \boldsymbol{z}) \leq \\ &f(\boldsymbol{a}[y_1 = \boldsymbol{b}(y_1), \ldots, y_m = \boldsymbol{b}(y_m)] \sqcup \boldsymbol{z}) \\ &\quad \Longleftrightarrow \\ &0 \leq g_{\boldsymbol{z}}(Y) - h_{\boldsymbol{z}}(Y) \end{aligned} \tag{8.6}$$

can be verified to hold for every $Y \subseteq J$ in time polynomial in n as the RHS of (8.6) is a submodular set function in Y. To verify the inequality we find the minimum value of the RHS of (8.6) for and compare it to 0. This can be done in time polynomial in n by one of the polynomial time algorithms for submodular function minimisation (see, e.g., [73, 87, 133] for descriptions of these algorithms). As $|Z| = |\mathcal{M}|^k$ we can perform this minimisation for every $\boldsymbol{z} \in Z$ within the stated time bound. Conversely, if (8.6) does not hold for some $Y \subseteq J$ and $x \in X$, then there is a tuple $\boldsymbol{t} \sqsubseteq \boldsymbol{b}$ such that $\boldsymbol{x}(\boldsymbol{t}) \not\leq f(\boldsymbol{t})$.

We will now show that it is in fact sufficient to verify the inequality for the tuples $\boldsymbol{y}$ such that $\boldsymbol{a} \sqsubseteq \boldsymbol{y} \sqsubseteq \boldsymbol{b}$. Let $\boldsymbol{y} \in \mathcal{M}^n$ be a tuple such that $\boldsymbol{y} \sqsubseteq \boldsymbol{b}$. Note that if $\boldsymbol{a} \sqsubseteq \boldsymbol{y}$, then it follows from (8.6) that $\boldsymbol{x}(\boldsymbol{y}) \leq f(\boldsymbol{y})$. For the sake of contradiction, assume that $\boldsymbol{x}(\boldsymbol{y}) \not\leq f(\boldsymbol{y})$. By the submodularity of f we get

$$f(\boldsymbol{a} \sqcup \boldsymbol{y}) + f(\boldsymbol{a} \sqcap \boldsymbol{y}) \leq f(\boldsymbol{a}) + f(\boldsymbol{y}). \tag{8.7}$$

As $\boldsymbol{a} \sqsubseteq \boldsymbol{a} \sqcup \boldsymbol{y} \sqsubseteq \boldsymbol{b}$ it follows from (8.6) that $\boldsymbol{x}(\boldsymbol{a} \sqcup \boldsymbol{y}) \leq f(\boldsymbol{a} \sqcup \boldsymbol{y})$. Furthermore, $\boldsymbol{y} \sqcap \boldsymbol{a} \sqsubseteq \boldsymbol{a}$ so by the assumptions in the lemma $\boldsymbol{x}(\boldsymbol{a} \sqcap \boldsymbol{y}) \leq f(\boldsymbol{a} \sqcap \boldsymbol{y})$. By the choice of $\boldsymbol{a}$ and $\boldsymbol{y}$ we get $\boldsymbol{x}(\boldsymbol{a}) = f(\boldsymbol{a})$ and $f(\boldsymbol{y}) < \boldsymbol{x}(\boldsymbol{y})$. It follows that

$$\boldsymbol{x}(\boldsymbol{a} \sqcup \boldsymbol{y}) + \boldsymbol{x}(\boldsymbol{a} \sqcap \boldsymbol{y}) \leq f(\boldsymbol{a} \sqcup \boldsymbol{y}) + f(\boldsymbol{a} \sqcap \boldsymbol{y}) \tag{8.8}$$

and

$$f(\boldsymbol{a}) + f(\boldsymbol{y}) < \boldsymbol{x}(\boldsymbol{a}) + \boldsymbol{x}(\boldsymbol{y}). \tag{8.9}$$

But

$$\boldsymbol{x}(\boldsymbol{a}) + \boldsymbol{x}(\boldsymbol{y}) \leq \boldsymbol{x}(\boldsymbol{a} \sqcup \boldsymbol{y}) + \boldsymbol{x}(\boldsymbol{a} \sqcap \boldsymbol{y}) \tag{8.10}$$

so we get a contradiction by combining (8.8), (8.7), (8.9), and (8.10). □

Given a submodular function f, it is not hard to see that the characteristic cone of $P_M(f)$ are the vectors $\boldsymbol{x} \in \mathbb{R}^{[n] \times A}$ such that $\boldsymbol{x} \leq 0$. Furthermore, the lineality space of $P_M(f)$ is $\{\mathbf{0}\}$. Given a polyhedron P such that lin.space $P = \{\mathbf{0}\}$, it is well-known (see, e.g., [132, Chapter 8]) that any $\boldsymbol{x} \in P$ can be represented as $\boldsymbol{x} = \sum_{i=1}^{n+1} \lambda_i \boldsymbol{y_i} + \boldsymbol{c}$ where $\boldsymbol{y_1}, \ldots, \boldsymbol{y_{n+1}}$ are vertices of P, $\boldsymbol{c} \in$ char.cone P, $\sum_{i=1}^{n+1} \lambda_i = 1$, and $\lambda_i \geq 0$ for all i. (That is, $\boldsymbol{x}$ is the sum of a convex combination of some of the vertices of P and a

vector in the characteristic cone of P.) The fact that $n+1$ vertices suffice is also well-known and is a corollary to Carathéodory's Theorem [29] (see [132, Chapter 7.7] for a proof of the theorem and the corollary). We state this result adapted to our setting as the following theorem.

Theorem 8.16. *Let $f : \mathcal{M}^n \to \mathbb{R}$ be submodular and let $\boldsymbol{x} \in P_M(f)$. Let $N = n \cdot |A|$. There are vertices $\boldsymbol{y_1}, \ldots, \boldsymbol{y_{N+1}}$ of $P_M(f)$, coefficients $\lambda_1, \ldots, \lambda_{N+1} \in \mathbb{R}$, and a vector $\boldsymbol{c} \in \mathbb{R}^{[n]\times A}$ such that*

$$\boldsymbol{x} = \sum_{i=1}^{N+1} \lambda_i \boldsymbol{y_i} + \boldsymbol{c},$$

$\boldsymbol{c} \leq 0$, $\sum_{i=1}^{N+1} \lambda_i = 1$, and $\lambda_i \geq 0$ for each $i \in [N+1]$.

We continue by showing that the vertices of $P_M(f)$ can be encoded in not too many bits. This is needed in the proof of Theorem 8.19.

Lemma 8.17 (Part of Lemma 6.2.4 in [73])**.** *Let $P \subseteq \mathbb{R}^m$ be a polyhedron. If P has facet-complexity at most ϕ, then P has vertex-complexity at most $4m^2\phi$.*

Lemma 8.18. *There is a constant c such that for any submodular $f : \mathcal{M}^n \to \mathbb{Z}$ the polyhedron $P_M(f)$ has vertex-complexity at most*

$$c \cdot |A| n^3 \cdot \log \max(|f|).$$

Proof. From the definition of $P_M(f)$ it follows that $P_M(f)$ has facet-complexity at most $c \cdot |A| n \cdot \log \max(|f|)$, for some constant c. The lemma now follows from Lemma 8.17. □

Lemma 8.18 tells us that the vertices of $P_M(f)$ can be encoded with not too many bits (that is, the size is bounded by a polynomial in n and $\log \max(|f|)$). We are now ready to prove the main theorem in this section, that $\mathcal{M}$ is well-characterised.

Theorem 8.19. *For every $k \geq 3$ the lattice $\mathcal{M}_k$ is well-characterised.*

As usual we let $\mathcal{M}$ denote an arbitrary diamond. The idea in the proof is that any point in $P_M(f)$ can be represented as a convex combination of at most $n|A| + 1$ vertices of $P_M(f)$ (this is Carathéodory's theorem). Furthermore, by Lemma 8.14 and an iterated use of Lemma 8.15 there are membership proofs for the vertices of $P_M(f)$ which can be checked in polynomial time. Hence, we get membership proofs for all of $P_M(f)$ which can be checked efficiently and by Theorem 8.5 we obtain the result.

Proof. Let $f : \mathcal{M}^n \to \mathbb{Z}$ be a submodular function and let m be some integer. We will show that if $\min_{\boldsymbol{t} \in \mathcal{M}^n} f(\boldsymbol{t}) = m$, then there is a proof of this fact which can be checked in time polynomial in n.

We can assume that $f(\mathbf{0}_{\mathcal{M}^n}) = 0$ as $\boldsymbol{t} \mapsto f(\boldsymbol{t}) - f(\mathbf{0}_{\mathcal{M}^n})$ is submodular. Let $N = n \cdot |A|$. The proof consists of a tuple $\boldsymbol{m} \in \mathcal{M}^n$, $N+1$ vectors

$\boldsymbol{x_1}, \ldots, \boldsymbol{x_{N+1}} \in \mathbb{R}^{[n]\times A}$, for each $i \in [N+1]$ a sequence $\boldsymbol{t_i^1}, \ldots, \boldsymbol{t_i^{2n}} \in \mathcal{M}^n$ of tuples, and finally an integer-valued vector $\boldsymbol{c} \in \mathbb{R}^{[n]\times A}$. To verify the proof we first find $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{N+1}) \in \mathbb{R}^{N+1}$ and $\boldsymbol{y} \in \mathbb{R}^{[n]\times A}$ such that

$$\boldsymbol{y} \leq 0, \quad \sum_{i=1}^{N+1} \lambda_i \boldsymbol{x_i} + \boldsymbol{y} = \boldsymbol{c}, \quad \boldsymbol{\lambda} \geq 0, \quad \text{and} \quad \sum_{i=1}^{N+1} \lambda_i = 1. \tag{8.11}$$

This can be done in time polynomial in n. Reject the proof if there are no solutions to (8.11). We proceed by checking that for each i

- $\boldsymbol{0_{\mathcal{M}^n}} = \boldsymbol{t_i^1} \sqsubseteq \boldsymbol{t_i^2} \sqsubseteq \ldots \sqsubseteq \boldsymbol{t_i^{2n-1}} \sqsubseteq \boldsymbol{t_i^{2n}} = \boldsymbol{1_{\mathcal{M}^n}}$, and
- $\boldsymbol{t_i^1}, \ldots, \boldsymbol{t_i^{2n}}$ are $\boldsymbol{x_i}$-tight, and
- for any $j \in [2n-1]$ there it at most one coordinate $l \in [n]$ such that $\boldsymbol{t_i^j}(l) = 0_{\mathcal{M}}$ and $\boldsymbol{t_i^{j+1}}(l) = 1_{\mathcal{M}}$.

Reject the proof if any of these checks fail. We now want to verify that $\boldsymbol{x_i} \in P_M(f)$, this can be done by using Lemma 8.15 repeatedly. For $j = 1, 2, \ldots, 2n-1$ we use the algorithm in Lemma 8.15 with $\boldsymbol{a} = \boldsymbol{t_i^j}$ and $\boldsymbol{b} = \boldsymbol{t_i^{j+1}}$. If all invocations of the algorithm succeed we can conclude that $\boldsymbol{x_i} \in P_M(f)$, otherwise the proof is rejected. Finally, compute

$$\boldsymbol{c} = \sum_{i=1}^{N+1} \lambda_i \boldsymbol{x_i} + \boldsymbol{y}$$

and accept the proof if $\boldsymbol{c} \leq 0$, $\boldsymbol{c}$ is unified, and $\boldsymbol{c}(\boldsymbol{1_{\mathcal{M}^n}}) = f(\boldsymbol{m}) = m$.

We now prove that this proof system is sound and complete.

Completeness (That is, if $m = \min_{\boldsymbol{y} \in \mathcal{M}^n} f(\boldsymbol{y})$, then there is a proof which the verifier accepts.) By Theorem 8.5 there is a unified integer-valued vector $\boldsymbol{c}$ such that $\boldsymbol{c} \in P_M(f)$, $\boldsymbol{c} \leq 0$ and $m = \boldsymbol{c}(\boldsymbol{1_{\mathcal{M}^n}})$. By Theorem 8.16 there are vectors $\boldsymbol{x_1}, \ldots, \boldsymbol{x_{N+1}}$ such that for each $i \in [N+1]$ $\boldsymbol{x_i}$ is a vertex of $P_M(f)$ and $\boldsymbol{c}$ is the sum of a convex combination of $\boldsymbol{x_1}, \ldots, \boldsymbol{x_{N+1}}$ with coefficients $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{N+1})$ and some vector $\boldsymbol{y} \in \text{char.cone } P_M(f)$, hence $\boldsymbol{\lambda}, \boldsymbol{y}$ is a solution to (8.11).

As for each $i \in [N+1]$ the vector $\boldsymbol{x_i}$ is a vertex of $P_M(f)$ it follows from Lemma 8.14 (and the observation that $\boldsymbol{0_{\mathcal{M}^n}}$ and $\boldsymbol{1_{\mathcal{M}^n}}$ are $\boldsymbol{x_i}$-tight[1]) that there is a sequence of tuples $\boldsymbol{t_i^1}, \boldsymbol{t_i^2}, \ldots, \boldsymbol{t_i^{2n}}$ such that $\boldsymbol{0_{\mathcal{M}^n}} = \boldsymbol{t_i^1} \sqsubseteq \boldsymbol{t_i^2} \sqsubseteq \ldots \sqsubseteq \boldsymbol{t_i^{2n}} = \boldsymbol{1_{\mathcal{M}^n}}$ and for each $j \in [2n-1]$ there is at most one $l \in [n]$ such

[1] To show that $\boldsymbol{1_{\mathcal{M}^n}}$ is $\boldsymbol{x_i}$-tight one can use the same technique as in Lemma 8.14: as $\boldsymbol{x_i}$ is a vertex there is some $\boldsymbol{c} \in \mathbb{R}^{[n]\times A}$ such that $\boldsymbol{x_i}$ is the unique solution to $\max\langle \boldsymbol{c}, \boldsymbol{y} \rangle, \boldsymbol{y} \in P_M(f)$. It follows that $\boldsymbol{c}(j,a) > 0$ for each $j \in [n]$ and $a \in A$ (otherwise $\boldsymbol{x_i}$ is either not the unique optimum or it is not an optimum at all). Hence, for each $j \in [n]$ and $a \in A$ if we try to increase $\boldsymbol{x_i}(j,a)$ then we get a vector which is not contained in $P_M(f)$. It follows that there is an $\boldsymbol{x_i}$-tight tuple $\boldsymbol{t_{j,a}}$ such that $a \sqsubseteq \boldsymbol{t_{j,a}}(j)$. By Lemma 8.2 we now get that $\boldsymbol{1_{\mathcal{M}^n}}$ is $\boldsymbol{x}$-tight.

that $\boldsymbol{t_i^j}(l) = 0_{\mathcal{M}}$ and $\boldsymbol{t_i^{j+1}}(l) = 1_{\mathcal{M}}$. It follows that this proof is accepted by the verifier. □

Soundness (That is, if there is a proof which the verifier accepts, then $m = \min_{\boldsymbol{y} \in \mathcal{M}^n} f(\boldsymbol{y})$.) As the verifier accepted the proof it follows from Lemma 8.15 that $\boldsymbol{x_i} \in P_M(f)$ for each $i \in [N+1]$. As $\boldsymbol{\lambda}$ and $\boldsymbol{y}$ is a solution to (8.11) it follows that $\boldsymbol{c} \in P_M(f)$ (it is a sum of a convex combination of some vectors contained in $P_M(f)$ and a vector in char.cone $P_M(f)$). From the acceptance of the verifier it also follows that $\boldsymbol{c} \geq 0$, $\boldsymbol{c}$ is unified, and $m = f(\boldsymbol{m}) = \boldsymbol{c}(\boldsymbol{1_{\mathcal{M}^n}})$. It now follows from Theorem 8.5 that $m = \min_{\boldsymbol{y} \in \mathcal{M}^n} f(\boldsymbol{y})$. □

In the proof system above, instead of letting the verifier solve (8.11) we could have required that $\boldsymbol{\lambda}$ and $\boldsymbol{y}$ are given in the proof. However, it is not obvious that $\boldsymbol{\lambda}$ and $\boldsymbol{y}$ can be encoded in polynomially many bits. This follows from the approach taken above by the fact that there are polynomial-time algorithms for finding solutions to systems of linear inequalities and Lemma 8.18.

Note that the vectors given in the proof do not need to be vertices of $P_M(f)$. However, by using the tight tuples and by repeatedly using Lemma 8.15 we can verify that the given vectors are in fact contained in $P_M(f)$ anyway. By Lemma 8.14 vectors and tight tuples always exist which satisfies the conditions above (namely, if we chose some appropriate vertices of $P_M(f)$).

The following lemma, which uses Lemma 8.15 essentially as we use it in Theorem 8.19, will be useful to us in Section 8.7.

Lemma 8.20. *Let k be some fixed positive integer. Let $f : \mathcal{M}^n \to \mathbb{Z}$ be submodular and let $\boldsymbol{x}$ be a vector in $P_M(f)$. Let $\boldsymbol{t_1}, \ldots, \boldsymbol{t_m} \in \mathcal{M}^n$ be $\boldsymbol{x}$-tight tuples such that $\boldsymbol{0_{\mathcal{M}^n}} = \boldsymbol{t_1} \sqsubset \ldots \sqsubset \boldsymbol{t_m} = \boldsymbol{1_{\mathcal{M}^n}}$ and for each $j \in [m-1]$ there is at most k distinct $i_1, i_2, \ldots, i_k \in [n]$ such that $\boldsymbol{t_j}(i_1) = \boldsymbol{t_j}(i_2) = \ldots = \boldsymbol{t_j}(i_k) = 0_{\mathcal{M}}$ and $\boldsymbol{t_{j+1}}(i_1) = \boldsymbol{t_{j+1}}(i_2) = \ldots = \boldsymbol{t_{j+1}}(i_k) = 1_{\mathcal{M}}$.*

For $i \in [m]$ let $E_i \subseteq I(\boldsymbol{t_i})$ such that $\boldsymbol{e} \in E_i$ if and only if $\langle \boldsymbol{e}, \boldsymbol{x} \rangle = f(\boldsymbol{t_i})$. Given $\boldsymbol{c} \in \mathbb{Q}^{[n] \times A}$, $\boldsymbol{x}$, and $\boldsymbol{t_1}, \ldots, \boldsymbol{t_m}$ it is possible to compute $\max\langle \boldsymbol{c}, \boldsymbol{y} \rangle$ subject to $\boldsymbol{y} \in P_M(f)$ and $\langle \boldsymbol{e}, \boldsymbol{y} \rangle = f(\boldsymbol{t_i})$ for all $i \in [m]$ and $\boldsymbol{e} \in E_i$ in time polynomial in n, $\log \max(|f|)$ and the encoding length of $\boldsymbol{c}$.

Note that we do not require that the running time depend polynomially on k.

Proof. We construct a separation algorithm for the polyhedron

$$\{\boldsymbol{y} \in P_M(f) \mid \forall i \in [m], \boldsymbol{e} \in E_i : \langle \boldsymbol{e}, \boldsymbol{y} \rangle = f(\boldsymbol{t_i})\}. \tag{8.12}$$

The lemma then follows from the equivalence of separation and optimisation given by the Ellipsoid algorithm.

Given a vector $\boldsymbol{y} \in \mathbb{R}^{[n] \times A}$ we first test that for all $i \in [m]$ and $\boldsymbol{e} \in E_i$ we have $\langle \boldsymbol{e}, \boldsymbol{y} \rangle = f(\boldsymbol{t_i})$. For each $i \in [m]$ we do this as follows: for each $j \in [n]$ such that $\boldsymbol{t_i}(j) = 1_{\mathcal{M}}$ the set of pairs of atoms $a, b \in A$ such that $\boldsymbol{x}(j, a) + \boldsymbol{x}(j, b)$ is maximised must be a subset of the set of pairs of atoms

$a', b' \in A$ such that $\boldsymbol{y}(j, a') + \boldsymbol{y}(j, b')$ is maximised. (Otherwise there is some $\boldsymbol{e} \in E_i$ such that $\langle \boldsymbol{x}, \boldsymbol{e} \rangle = f(\boldsymbol{t_i}) \neq \langle \boldsymbol{y}, \boldsymbol{e} \rangle$.) If this is the case then $\langle \boldsymbol{y}, \boldsymbol{e} \rangle = f(\boldsymbol{t_i})$ for *all* $\boldsymbol{e} \in E_i$ if and only if $\langle \boldsymbol{y}, \boldsymbol{e} \rangle = f(\boldsymbol{t_i})$ for *some* $\boldsymbol{e} \in E_i$.

Note that this test can be done in polynomial time in n and $\log \max(|f|)$ as $m \leq |A| \cdot n$. We can then use the algorithm in Lemma 8.15 to test if $\boldsymbol{y} \in P_M(f)$. By combining these two tests we have a separation oracle for (8.12) and hence the lemma follows. □

8.7 Finding the Minimum Value

In this section we will show that there is an algorithm which finds the minimum value of a submodular $f : \mathcal{M}^n \to \mathbb{Z}$ in time polynomial in n and $\max(|f|)$. Note that from an algorithm which computes $\min_{\boldsymbol{t} \in \mathcal{M}^n} f(\boldsymbol{t})$ one can construct an algorithm to find a minimiser of f, i.e., find a tuple $\boldsymbol{y} \in \mathcal{M}^n$ such that $f(\boldsymbol{y}) = \min_{\boldsymbol{t} \in \mathcal{M}^n} f(\boldsymbol{t})$. This can be done by for each $x \in \mathcal{M}$ minimising $f_x : \mathcal{M}^{n-1} \to \mathbb{R}$ defined by $f_x(\boldsymbol{t}) = f(x, \boldsymbol{t})$. If $\min_{\boldsymbol{t} \in \mathcal{M}^{n-1}} f_x(\boldsymbol{t}) = \min_{\boldsymbol{t} \in \mathcal{M}^n} f(\boldsymbol{t})$, then there is a minimiser $\boldsymbol{y} \in \mathcal{M}^n$ to f such that $\boldsymbol{y}(1) = x$. By iterating this procedure n times one finds a minimiser of f.

We start with a high-level description of the algorithm. The starting point is the separation problem for $P_M(f)$ and the observation that $\mathbf{0} \in P_M(f)$ if and only if $\min_{\boldsymbol{t} \in \mathcal{M}^n} f(\boldsymbol{t}) \geq 0$. Hence, given an algorithm for deciding if $\mathbf{0}$ is contained in $P_M(f)$ we can apply a binary search strategy to find a minimiser of f.

In each iteration i of the algorithm we maintain an upper bound u_i and lower bound l_i on $\min_{\boldsymbol{t} \in \mathcal{M}^n} f(\boldsymbol{t})$. If $\mathbf{0} \in P_M(f - (u_i - l_i)/2)$ (note that for any $c \in \mathbb{R}$ the function $f + c$ is submodular if f is submodular), we iterate the algorithm with $u_{i+1} = u_i$ and $l_{i+1} = (u_i - l_i)/2$. Otherwise, if $\mathbf{0} \notin P_M(f)$, we set $u_{i+1} = (u_i - l_i)/2$ and $l_{i+1} = l_i$. For an initial upper bound we can use $u_1 = f(\mathbf{0}_{\boldsymbol{\mathcal{M}^n}})$. To find an initial lower bound l_1 we can use Theorem 8.7 together with the greedy algorithm in Lemma 8.3. The running time of this algorithm is $O(S \cdot \log \max(|f|) + n)$, where S is the time taken to decide if $\mathbf{0} \in P_M(f)$.

By the equivalence of separation and optimisation given by the Ellipsoid algorithm (Theorem 8.13) it is sufficient to solve the optimisation problem for $P_M(f)$. In the optimisation problem we are given $\boldsymbol{c} \in \mathbb{Q}^{[n] \times A}$ and are supposed to maximise $\langle \boldsymbol{c}, \boldsymbol{y} \rangle$ subject to $\boldsymbol{y} \in P_M(f)$. To get the running time we are aiming for we must do this in time polynomial in n, $\max(|f|)$ and the encoding length of $\boldsymbol{c}$.

To solve this problem our algorithm starts with a vertex of $P_M(f)$ and either finds an adjacent vertex with a strictly better measure or concludes that no such vertex exists. The initial vertex is found by the greedy algorithm in Lemma 8.3. To make this approach run in pseudopolynomial time two parts are needed. The first one is the existence of a pseudopolynomial-time algorithm to go from a vertex to a better one or conclude that no such

vertex exists. We present an algorithm for this in Section 8.7.3. The other part is that we must ensure that the algorithm makes enough progress in each iteration so that we get the bound on the running time we are aiming for. To this end, we prove in Section 8.7.2 that the vertices of $P_M(f)$ are half-integral (i.e., every coordinate of every vertex of $P_M(f)$ is contained in $\{1/2 \cdot k \mid k \in \mathbb{Z}\}$).

This section is organised as follows: in Subsection 8.7.1 we prove a couple of results of the structure of the vertices of $P_M(f)$. We also show that a submodular function can be turned into a strictly submodular function such that any minimiser of the latter is also a minimiser of the former. This will be useful to us in the subsequent parts of the algorithm. In Subsection 8.7.2 we prove that the vertices of $P_M(f)$ are half-integral. Finally, in Subsection 8.7.3 we show how we can go from one vertex of $P_M(f)$ to a better one (if there is one) and how this can be used to construct an optimisation algorithm for $P_M(f)$. The oracle-pseudo-tractability for the diamonds is stated as Corollary 8.31.

8.7.1 The Structure of the Vertices of $P_M(f)$

The following lemma is stated in [134, Theorem 2.1] for the boolean lattice. Essentially the same proof works for modular lattices. We give a version of the lemma specialised to $\mathcal{M}^n$.

Lemma 8.21. *Let $\boldsymbol{t}, \boldsymbol{u} \in \mathcal{M}^n$ such that $\boldsymbol{t} \not\sqsubseteq \boldsymbol{u}$ and $\boldsymbol{u} \not\sqsubseteq \boldsymbol{t}$, then*

$$\begin{aligned}&\rho(\boldsymbol{t})(2n - \rho(\boldsymbol{t})) + \rho(\boldsymbol{u})(2n - \rho(\boldsymbol{u})) \quad > \\ &\rho(\boldsymbol{t} \sqcap \boldsymbol{u})(2n - \rho(\boldsymbol{t} \sqcap \boldsymbol{u})) + \rho(\boldsymbol{t} \sqcup \boldsymbol{u})(2n - \rho(\boldsymbol{t} \sqcup \boldsymbol{u})).\end{aligned}$$

Proof. Let $\alpha = \rho(\boldsymbol{t} \sqcap \boldsymbol{u}), \beta = \rho(\boldsymbol{t}) - \rho(\boldsymbol{t} \sqcap \boldsymbol{u})$, $\gamma = \rho(\boldsymbol{u}) - \rho(\boldsymbol{t} \sqcap \boldsymbol{u})$, and $\delta = 2n - \rho(\boldsymbol{t} \sqcup \boldsymbol{u})$. Then the LHS is equal to

$$(\alpha + \beta)(\gamma + \delta) + (\alpha + \gamma)(\beta + \delta) = 2\alpha\delta + 2\beta\gamma + \alpha\gamma + \beta\delta + \alpha\beta + \gamma\delta$$

as ρ is modular (i.e., $\rho(\boldsymbol{t}) + \rho(\boldsymbol{u}) = \rho(\boldsymbol{t} \sqcup \boldsymbol{u}) + \rho(\boldsymbol{t} \sqcap \boldsymbol{u})$). The RHS is equal to

$$\alpha(\beta + \gamma + \delta) + (\alpha + \beta + \gamma)\delta = 2\alpha\delta + \alpha\gamma + \beta\delta + \alpha\beta + \gamma\delta.$$

Since $\beta\gamma > 0$ the lemma follows. □

The lemma above tells us that the function $\boldsymbol{t} \mapsto \rho(\boldsymbol{t})(2n - \rho(\boldsymbol{t}))$ is strictly submodular. Note that if $f : \mathcal{M}^n \to \mathbb{R}$ is submodular, then f can be turned into a strictly submodular function $f' : \mathcal{M}^n \to \mathbb{R}$ by $f'(\boldsymbol{t}) = f(\boldsymbol{t}) + \epsilon\rho(\boldsymbol{t})(2n - \rho(\boldsymbol{t}))$ with a small enough $\epsilon > 0$. Observe that if $\epsilon > 0$ is chosen small enough then any minimiser of f' is also a minimiser of f. Strictly submodular functions are an interesting subset of the submodular functions due to this observation and the following lemma.

Lemma 8.22. *Let $f : \mathcal{M}^n \to \mathbb{R}$ be strictly submodular and let $\boldsymbol{x}$ be a vertex of $P_M(f)$. Then, the $\boldsymbol{x}$-tight tuples form a chain.*

Proof. Assume, for the sake of contradiction, that $\boldsymbol{t}, \boldsymbol{u} \in \mathcal{M}^n$ are $\boldsymbol{x}$-tight and $\boldsymbol{t} \not\sqsubseteq \boldsymbol{u}$ and $\boldsymbol{u} \not\sqsubseteq \boldsymbol{t}$. It follows that

$$\begin{aligned} \boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{x}(\boldsymbol{u}) &= f(\boldsymbol{t}) + f(\boldsymbol{u}) > f(\boldsymbol{t} \sqcap \boldsymbol{u}) + f(\boldsymbol{t} \sqcup \boldsymbol{u}) \\ &= \boldsymbol{x}(\boldsymbol{u} \sqcup \boldsymbol{v}) + \boldsymbol{x}(\boldsymbol{u} \sqcap \boldsymbol{v}). \end{aligned} \tag{8.13}$$

The last equality follows from the fact that the $\boldsymbol{x}$-tight tuples are closed under $\sqcap$ and $\sqcup$ (Lemma 8.2). However, (8.13) contradicts the supermodularity of $\boldsymbol{x}$. □

Lemma 8.22 tells us that, for strictly submodular f, for each vertex $\boldsymbol{x}$ of $P_M(f)$ the set of $\boldsymbol{x}$-tight tuples is a chain in $\mathcal{M}^n$. As the dimension of $P_M(f)$ is $|A|n$, for every vertex $\boldsymbol{x}$ of $P_M(f)$ there are $|A|n$ linearly independent inequalities which are satisfied with equality by $\boldsymbol{x}$. This means that for every such $\boldsymbol{x}$ there is a chain $\boldsymbol{t_1} \sqsubset \boldsymbol{t_2} \sqsubset \ldots \sqsubset \boldsymbol{t_m}$ in $\mathcal{M}^n$ and linearly independent vectors $\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots, \boldsymbol{e_{|A|n}}$ such that for each $i \in [|A|n]$ there is some $j(i) \in [m]$ such that $\boldsymbol{e_i} \in I(\boldsymbol{t_{j(i)}})$ and for all $i \in [|A|n]$ we have $\langle \boldsymbol{e_i}, \boldsymbol{x} \rangle = f(\boldsymbol{t_{j(i)}})$. Furthermore, $\boldsymbol{x}$ is the only vector which satisfies $\langle \boldsymbol{e_i}, \boldsymbol{x} \rangle = f(\boldsymbol{t_i})$ for all $i \in [|A|n]$.

For general (not necessarily strict) submodular functions the set of $\boldsymbol{x}$-tight tuples is not necessarily a chain, but one can prove that for every vertex there is a chain of tuples such that some subset of the inequalities induced by the tight tuples characterises the vertex. That is, given the subset of inequalities induced by such a chain of tight tuples there is only one point in $P_M(f)$ which satisfies all the inequalities with equality. For our purposes Lemma 8.22 suffices, but we state and prove this latter result as well in the following lemma.

Lemma 8.23. *Let $f : \mathcal{M}^n \to \mathbb{R}$ be submodular and let $\boldsymbol{x}$ be a vertex of $P_M(f)$. Then there is a chain $\boldsymbol{t_1} \sqsubset \boldsymbol{t_2} \sqsubset \ldots \sqsubset \boldsymbol{t_m}$ in $\mathcal{M}^n$ and linearly independent vectors $\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots, \boldsymbol{e_{|A|n}}$, which are all $\boldsymbol{x}$-tight, such that for each $i \in [|A|n]$ there is some $j(i) \in [m]$ such that $\boldsymbol{e_i} \in I(\boldsymbol{t_{j(i)}})$.*

Proof. Define $f'(\boldsymbol{t}) = f(\boldsymbol{t}) + \epsilon\rho(\boldsymbol{t})(2n - \rho(\boldsymbol{t}))$ and choose $\epsilon > 0$ small. From Lemma 8.21 it follows that f' is strictly submodular. Choose $\boldsymbol{c} \in \mathbb{R}^{[n] \times A}$ such that $\boldsymbol{x}$ is the unique optimum to $\max\langle \boldsymbol{c}, \boldsymbol{y} \rangle, \boldsymbol{y} \in P_M(f)$. Let $\boldsymbol{x'}$ be a vertex of $P_M(f')$ which is an optimum to $\max\langle \boldsymbol{c}, \boldsymbol{y} \rangle, \boldsymbol{y} \in P_M(f')$. From Lemma 8.22 it follows that as $\boldsymbol{x'}$ is a vertex of $P_M(f')$ there are $\boldsymbol{t_1} \sqsubset \boldsymbol{t_2} \sqsubset \ldots \sqsubset \boldsymbol{t_m}$ and $\boldsymbol{x'}$-tight $\boldsymbol{e_1}, \ldots, \boldsymbol{e_{n|A|}}$ as in the statement of the lemma. Let $\boldsymbol{e_1}, \ldots, \boldsymbol{e_{n|A|}}$ be the rows of the matrix A and define $\boldsymbol{b} = (f(\boldsymbol{t_{j(1)}}), \ldots, f(\boldsymbol{t_{j(m)}}))^T$ and

$$\boldsymbol{\epsilon} = \epsilon \cdot (\rho(\boldsymbol{t_{j(1)}})(2n - \rho(\boldsymbol{t_{j(1)}})), \ldots, \rho(\boldsymbol{t_{j(m)}})(2n - \rho(\boldsymbol{t_{j(m)}})))^T.$$

It follows that $\boldsymbol{x'} = A^{-1}(\boldsymbol{b} + \boldsymbol{\epsilon})$. We proceed by establishing two claims.

Claim A. $A^{-1}\boldsymbol{b} \in P_M(f)$**.**

Proof. To see this assume for the sake of contradiction that $A^{-1}\boldsymbol{b} \notin$

$P_M(f)$, then there is some $\boldsymbol{t} \in \mathcal{M}^n$ and $\boldsymbol{e} \in I(\boldsymbol{t})$ such that $\boldsymbol{e}A^{-1}\boldsymbol{b} > f(\boldsymbol{t})$. However,

$$\boldsymbol{e}\boldsymbol{x'} = \boldsymbol{e}A^{-1}(\boldsymbol{b}+\boldsymbol{\epsilon}) = \boldsymbol{e}A^{-1}\boldsymbol{b} + \boldsymbol{e}A^{-1}\boldsymbol{\epsilon} \leq f(\boldsymbol{t}) + \epsilon\rho(\boldsymbol{t})(2n-\rho(\boldsymbol{t})). \tag{8.14}$$

As $\boldsymbol{e}A^{-1}\boldsymbol{b} > f(\boldsymbol{t})$ we can choose some $\delta > 0$ such that $\boldsymbol{e}A^{-1}\boldsymbol{b} > f(\boldsymbol{t}) + \delta$. By choosing ϵ so that $|\epsilon\rho(\boldsymbol{t})(2n-\rho(\boldsymbol{t})) - \boldsymbol{e}A^{-1}\boldsymbol{\epsilon}| < \delta$ we get

$$\boldsymbol{e}A^{-1}\boldsymbol{b} + \boldsymbol{e}A^{-1}\boldsymbol{\epsilon} > f(\boldsymbol{t}) + \epsilon\rho(\boldsymbol{t})(2n-\rho(\boldsymbol{t}))$$

which contradicts (8.14). We conclude that $A^{-1}\boldsymbol{b} \in P_M(f)$.

Claim B. $\langle A^{-1}\boldsymbol{b}, \boldsymbol{c}\rangle = \langle \boldsymbol{x}, \boldsymbol{c}\rangle$**.**
Proof. Assume, for the sake of contradiction, that $\langle A^{-1}\boldsymbol{b}, \boldsymbol{c}\rangle < \langle \boldsymbol{x}, \boldsymbol{c}\rangle$. (The inequality $\leq$ follows from our choice of $\boldsymbol{x}$ and Claim A.) Note that $\boldsymbol{x} \in P_M(f) \subseteq P_M(f')$. For sufficiently small ϵ we get

$$\langle \boldsymbol{x'}, \boldsymbol{c}\rangle = \langle A^{-1}(\boldsymbol{b}+\boldsymbol{\epsilon}), \boldsymbol{c}\rangle < \langle \boldsymbol{x}, \boldsymbol{c}\rangle$$

which contradicts the optimality of $\boldsymbol{x'}$.

We have shown that for every vertex $\boldsymbol{x} \in P_M(f)$ there is some vertex $\boldsymbol{x'} \in P_M(f')$ which satisfies some inequalities, given by the matrix A, with equality. As f' is strictly submodular it follows from Lemma 8.22 that the $\boldsymbol{x'}$-tight tuples form a chain. By Claim A the inequalities in A also defines a point in $P_M(f)$. Furthermore, by Claim B this point maximises $\langle \boldsymbol{y}, \boldsymbol{c}\rangle$ over $P_M(f)$. By our choice of $\boldsymbol{c}$ it follows that $A^{-1}\boldsymbol{b} = \boldsymbol{x}$. The lemma follows. □

8.7.2 $P_M(f)$ is Half-integral

In this subsection we will prove that if $f : \mathcal{M}^n \to \mathbb{Z}$ is submodular, then the vertices of $P_M(f)$ are half-integral.

Lemma 8.24. *Let $f : \mathcal{M}^n \to \mathbb{R}$ be submodular and let $\boldsymbol{x}$ be a vertex of $P_M(f)$. For each $i \in [n]$ there are three possibilities*

1. *$\boldsymbol{x}(i,a) = \boldsymbol{x}(i,b)$ for all $a, b \in A$; or*
2. *there is exactly one atom $a' \in A$ such that $\boldsymbol{x}(i,a') > \min_{a\in A} \boldsymbol{x}(i,a)$; or*
3. *there is exactly one atom $a' \in A$ such that $\boldsymbol{x}(i,a') < \max_{a\in A} \boldsymbol{x}(i,a)$.*

Proof. As $\boldsymbol{x}$ is a vertex of $P_M(f)$ there is a $\boldsymbol{c} \in \mathbb{R}^{[n]\times A}$ such that $\boldsymbol{x}$ is the unique optimum to $\langle \boldsymbol{c}, \boldsymbol{y}\rangle, \boldsymbol{y} \in P_M(f)$. As the optimum exist it follows that $\boldsymbol{c} \geq \boldsymbol{0}$. Assume, for the sake of contradiction, that there is a coordinate $i \in [n]$ such that the statement of the lemma does not hold for i. Let A_1 be the atoms $a' \in A$ which satisfies $\boldsymbol{x}(i,a') = \max_{a\in A} \boldsymbol{x}(i,a)$. Similarly, let A_2

be the atoms $a' \in A_2$ which satisfies $\boldsymbol{x}(i,a') = \max_{a \in A \setminus A_1} \boldsymbol{x}(i,a)$. Finally, let $A_3 = A \setminus (A_1 \cup A_2)$. We will first prove the following claim.

Claim. If the statement of the lemma does not hold, then there are distinct atoms $b, c \in A$ and $\boldsymbol{x}$-tight tuples $\boldsymbol{t_1}, \boldsymbol{t_2} \in \mathcal{M}^n$ such that $\boldsymbol{t_1}(i) = b$, $\boldsymbol{t_2}(i) = c$ and ($b \in A_3$ or $b, c \in A_2$).

Proof. If $a \in A_3$ let $\boldsymbol{x'} = \boldsymbol{x} + \delta\boldsymbol{\chi}(i,a)$ for some small $\delta > 0$. As $\boldsymbol{x}$ is the unique optimum it follows that $\boldsymbol{x'} \notin P_M(f)$ and hence there is an $\boldsymbol{x}$-tight tuple $\boldsymbol{t} \in \mathcal{M}^n$ such that $\boldsymbol{t}(i) = a$. So if $|A_3| \geq 2$, then the claim holds. Similarly, if $|A_1| \geq 2$, then any for any $a \in A \setminus A_1$ we get an $\boldsymbol{x}$-tight tuple $\boldsymbol{t}$ such that $\boldsymbol{t}(i) = a$. (Again this follows from considering the vector $\boldsymbol{x'} = \boldsymbol{x} + \delta\boldsymbol{\chi}(i,a)$.)

So $|A_3| \leq 1$ and ($|A_1| = 1$ or $|A_1| \geq |A| - 1$). If $|A_3| = 0$ and ($|A_1| = 1$ or $|A_1| \geq |A| - 1$), then the statement of the lemma holds, so we must have $|A_3| = 1$. This implies that $|A_1| = 1$.

Let $A_1 = \{a\}$. If $\boldsymbol{c}(i,a) \geq \sum_{b \in A_2} \boldsymbol{c}(i,b)$, then let $\boldsymbol{x'} = \boldsymbol{x} + \delta\boldsymbol{\chi}(i,a) - \delta \sum_{b \in A_2} \boldsymbol{\chi}(i,b)$ for some small $\delta > 0$. It follows that $\boldsymbol{x'} \notin P_M(f)$ and hence there is an $\boldsymbol{x}$-tight tuple $\boldsymbol{t}$ with $\boldsymbol{t}(i) = a$. In the other case, when $\boldsymbol{c}(i,a) < \sum_{b \in A_2} \boldsymbol{c}(i,b)$, we let $\boldsymbol{x'} = \boldsymbol{x} - \delta\boldsymbol{\chi}(i,a) + \delta \sum_{b \in A_2} \boldsymbol{\chi}(i,b)$. It follows that there is an $\boldsymbol{x}$-tight tuple $\boldsymbol{t}$ with $\boldsymbol{t}(i) \in A_2$.

Let b and c be the atoms in the claim above and let $\boldsymbol{t_1}$ and $\boldsymbol{t_2}$ be the $\boldsymbol{x}$-tight tuples in the claim. As f is submodular we have

$$f(\boldsymbol{t_1} \sqcup \boldsymbol{t_2}) + f(\boldsymbol{t_1} \sqcap \boldsymbol{t_2}) \leq f(\boldsymbol{t_1}) + f(\boldsymbol{t_2}) = \boldsymbol{x}(\boldsymbol{t_1}) + \boldsymbol{x}(\boldsymbol{t_2}).$$

From this inequality and the fact that $\boldsymbol{x} \in P_M(f)$ it follows that

$$\begin{aligned}
&\boldsymbol{x}(\boldsymbol{t_1} \sqcup \boldsymbol{t_2}) + \boldsymbol{x}(\boldsymbol{t_1} \sqcap \boldsymbol{t_2}) && \leq \\
&f(\boldsymbol{t_1} \sqcup \boldsymbol{t_2}) + f(\boldsymbol{t_1} \sqcap \boldsymbol{t_2}) && \leq \\
&\boldsymbol{x}(\boldsymbol{t_1}) + \boldsymbol{x}(\boldsymbol{t_2}) && \leq \\
&\boldsymbol{x}(\boldsymbol{t_1} \sqcup \boldsymbol{t_2}) + \boldsymbol{x}(\boldsymbol{t_1} \sqcap \boldsymbol{t_2}).
\end{aligned}$$

We conclude that $\boldsymbol{x}(\boldsymbol{t_1} \sqcup \boldsymbol{t_2}) + \boldsymbol{x}(\boldsymbol{t_1} \sqcap \boldsymbol{t_2}) = \boldsymbol{x}(\boldsymbol{t_1}) + \boldsymbol{x}(\boldsymbol{t_2})$. However, this leads to a contradiction:

$$\begin{aligned}
&\boldsymbol{x}(\boldsymbol{t_1}) + \boldsymbol{x}(\boldsymbol{t_2}) = \boldsymbol{x}(i,b) + \boldsymbol{x}(i,c) + \boldsymbol{x}(\boldsymbol{t_1}[i = 0_{\mathcal{M}}]) + \boldsymbol{x}(\boldsymbol{t_2}[i = 0_{\mathcal{M}}]) && \leq \\
&\boldsymbol{x}(i,b) + \boldsymbol{x}(i,c) + \boldsymbol{x}((\boldsymbol{t_1} \sqcup \boldsymbol{t_2})[i = 0_{\mathcal{M}}]) + \boldsymbol{x}(\boldsymbol{t_1} \sqcap \boldsymbol{t_2}) && < \\
&\boldsymbol{x}(\boldsymbol{t_1} \sqcup \boldsymbol{t_2}) + \boldsymbol{x}(\boldsymbol{t_1} \sqcap \boldsymbol{t_2})
\end{aligned}$$

So the coordinate i cannot exist. □

The lemma above can be strengthened if $|A| = 3$, in this case only 1 and 2 are possible. To see this, assume that $A = \{a_1, a_2, a_3\}$ and $\boldsymbol{x}(i,a_1) = \boldsymbol{x}(i,a_2) > \boldsymbol{x}(i,a_3)$. Let $\boldsymbol{x'} = \boldsymbol{x} + \delta\boldsymbol{\chi}(i,a_1) - \delta\boldsymbol{\chi}(i,a_2)$ (or $\boldsymbol{x'} = \boldsymbol{x} - \delta\boldsymbol{\chi}(i,a_1) + \delta\boldsymbol{\chi}(i,a_2)$ if $\boldsymbol{c}(i,a_1) < \boldsymbol{c}(i,a_2)$). As $\boldsymbol{x'} \notin P_M(f)$ it follows that there is some $\boldsymbol{x}$-tight tuple $\boldsymbol{t}$ with $\boldsymbol{t}(i) = a_1$ (or $\boldsymbol{t}(i) = a_2$). We can then proceed as in the proof above.

We will need the following lemma from [77] (see also [3]) in our proof of the half-integrality of $P_M(f)$.

Lemma 8.25. *Let A be a $m \times n$ integral matrix satisfying*

$$\sum_{i=1}^{m} |A_{ij}| \leq 2$$

for $j \in [n]$. Then, for every square nonsingular submatrix S of A, S^{-1} is half-integral.

By combining Lemma 8.22, Lemma 8.24 and Lemma 8.25 we are able the obtain the following theorem which asserts the half-integrality of $P_M(f)$.

Theorem 8.26. *Let $f : \mathcal{M}^n \to \mathbb{Z}$ be submodular. For any vertex $\boldsymbol{x}$ of $P_M(f)$ and any $i \in [n]$ and $a \in A$ we have $\boldsymbol{x}(i,a) \in \{1/2 \cdot k \mid k \in \mathbb{Z}\}$.*

Proof. Let $\boldsymbol{x}$ be a vertex of $P_M(f)$. By Lemma 8.23 there is a chain of $\boldsymbol{x}$-tight tuples $\boldsymbol{t_1} \sqsubset \ldots \sqsubset \boldsymbol{t_m}$ and linearly independent vectors $\boldsymbol{e_1}, \ldots, \boldsymbol{e_{|A|n}}$ such that for each $i \in [|A|n]$ there is some $j(i) \in [m]$ such that $\boldsymbol{e_i} \in I(\boldsymbol{t_{j(i)}})$. We can also assume that for $i \leq i'$ we have $j(i) \leq j(i')$. If E is the matrix with rows $\boldsymbol{e_1}, \ldots, \boldsymbol{e_{|A|n}}$, then $\boldsymbol{x}$ is the unique solution to $E\boldsymbol{x} = \boldsymbol{b}$, where

$$\boldsymbol{b} = \left(f\left(\boldsymbol{t_{j(1)}}\right), f\left(\boldsymbol{t_{j(2)}}\right), \ldots, f\left(\boldsymbol{t_{j(|A|n)}}\right)\right)^T.$$

Let $A = \{a_1, a_2, \ldots, a_{|A|}\}$. By Lemma 8.24 we can assume, without loss of generality, that and $\boldsymbol{x}(i,a_2) = \boldsymbol{x}(i,a_3) = \ldots = \boldsymbol{x}(i,a_{|A|})$. If $\boldsymbol{x}(i,a_1) = \boldsymbol{x}(i,a_2) = \ldots = \boldsymbol{x}(i,a_{|A|})$, then we can identify $\boldsymbol{x}(i,a_1), \ldots, \boldsymbol{x}(i,a_{|A|})$ without changing the set of solutions to $E\boldsymbol{x} = \boldsymbol{b}$. In the other cases, when $\boldsymbol{x}(i,a_1) > \min_{a \in A} \boldsymbol{x}(i,a)$ or $\boldsymbol{x}(i,a_1) < \max_{a \in A} \boldsymbol{x}(i,a)$, we can identify $\boldsymbol{x}(i,a_2), \ldots, \boldsymbol{x}(i,a_{|A|})$ without changing the set of solutions to $E\boldsymbol{x} = \boldsymbol{b}$. After having identified these variables we get a system of linear equations, $E'\boldsymbol{x'} = \boldsymbol{b'}$, which has a unique solution. Furthermore, the solution to $E'\boldsymbol{x'} = \boldsymbol{b'}$ is half-integral if and only if $E\boldsymbol{x} = \boldsymbol{b}$ has a half-integral solution (that is, if and only if $\boldsymbol{x}$ is half-integral). Let $X \subseteq [n] \times \{1,2\}$ such that for each $i \in [n]$, $(i,1) \in X$ and $((i,2) \in X$ if and only if $(\boldsymbol{x}(i,a_1) > \min_{a \in A} \boldsymbol{x}(i,a)$ or $\boldsymbol{x}(i,a_1) < \max_{a \in A} \boldsymbol{x}(i,a)))$. We can describe the rows, $\boldsymbol{e'_1}, \boldsymbol{e'_2}, \ldots, \boldsymbol{e'_{|A|n}} \in \mathbb{R}^X$ of E' as follows

- if $\boldsymbol{x}(i,a_1) = \boldsymbol{x}(i,a_2) = \ldots = \boldsymbol{x}(i,a_{|A|})$, then $\boldsymbol{e'_j}(i,1) = \sum_{a \in A} \boldsymbol{e_j}(i,a)$;
- otherwise (if $\boldsymbol{x}(i,a_1) > \min_{a \in A} \boldsymbol{x}(i,a)$ or $\boldsymbol{x}(i,a_1) < \max_{a \in A} \boldsymbol{x}(i,a)$), then $\boldsymbol{e'_j}(i,1) = \boldsymbol{e_j}(i,a_1)$ and $\boldsymbol{e'_j}(i,2) = \sum_{a \in A, a \neq a_1} \boldsymbol{e'_j}(a)$.

As the solution to $E'\boldsymbol{x'} = \boldsymbol{b'}$ and $E\boldsymbol{x} = \boldsymbol{b}$ are equal, modulo the identification of some of the variables, there is a subset $R = \{r_1, r_2, \ldots, r_{|X|}\} \subseteq [|A|n]$ with $r_1 < r_2 < \ldots < r_{|X|}$ such that the matrix E'', with rows $\{\boldsymbol{e'_i} \mid i \in R\}$, has an inverse. Furthermore, this inverse is half-integral (that is, E''^{-1} is half-integral) if and only if the solution to $E'\boldsymbol{x'} = \boldsymbol{b'}$ is half-integral.

It is easy to see that the entries of E'' are contained in $\{0,1,2\}$. Furthermore, if $\boldsymbol{c}$ is an arbitrary column of E'', then it is of the form $(0, \ldots, 0,$

$1, \ldots, 1, 2, \ldots, 2)^T$ or $(0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0)^T$ (in these patterns, $x, \ldots, x$ means that x occurs zero or more times). It follows that for each $(i, k) \in X$ we have

$$\sum_{l=1}^{|X|} |\boldsymbol{e'}_{\boldsymbol{r}_{l+1}}(i,k) - \boldsymbol{e'}_{\boldsymbol{r}_l}(i,k)| \leq 2. \tag{8.15}$$

Following the proof of Theorem 1 in [67] we now define

$$U = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -1 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix}.$$

We can then express the inverse of E'' as $(UE'')^{-1}U$. From (8.15) and Lemma 8.25 it follows that $(UE'')^{-1}$ is half-integral and hence E''^{-1} is half-integral as well, which implies that $\boldsymbol{x}$ is half-integral. □

8.7.3 Finding Augmentations

Let $f : \mathcal{M}^n \to \mathbb{Z}$ be submodular. In this section we will show that there is an algorithm which decides if $\mathbf{0} \in P_M(f)$ in time polynomial in n and $\max(|f|)$. The strategy of the algorithm is to use the equivalence between separation and optimisation given by the Ellipsoid algorithm and solve the optimisation problem for $P_M(f)$ instead. In the optimisation problem we are given $\boldsymbol{c} \in \mathbb{Q}^{[n]\times A}$ and are supposed to find $\max\langle \boldsymbol{c}, \boldsymbol{y}\rangle, \boldsymbol{y} \in P_M(f)$. This problem is solved by iterating an augmentation step in which we are in some vertex $\boldsymbol{x}$ of $P_M(f)$ and wish to find some vertex $\boldsymbol{x'}$, adjacent to $\boldsymbol{x}$, such that $\langle \boldsymbol{c}, \boldsymbol{x'}\rangle > \langle \boldsymbol{c}, \boldsymbol{x}\rangle$.

Let $\boldsymbol{c} \in \mathbb{Q}^{[n]\times A}$ and assume that we want to solve $\max\langle \boldsymbol{c}, \boldsymbol{y}\rangle, \boldsymbol{y} \in P_M(f)$. Let T be the set of all $\boldsymbol{x}$-tight tuples and let $E \subseteq \cup_{\boldsymbol{t}\in T} I(\boldsymbol{t})$ such that $\boldsymbol{e} \in E$ if and only if there is some $\boldsymbol{t} \in T$ with $\boldsymbol{e} \in I(\boldsymbol{t})$ and $\langle \boldsymbol{e}, \boldsymbol{x}\rangle = f(\boldsymbol{t})$. Finding a vector $\boldsymbol{z} \in \mathbb{R}^{[n]\times A}$ such that there is some $\delta > 0$ which satisfies $\langle \boldsymbol{c}, \boldsymbol{x}\rangle < \langle \boldsymbol{c}, \boldsymbol{x} + \delta\boldsymbol{z}\rangle$ and $\boldsymbol{x} + \delta\boldsymbol{z} \in P_M(f)$ or conclude that no such vector $\boldsymbol{z}$ exists is equivalent to solving the linear program

$$\max\langle \boldsymbol{c}, \boldsymbol{z}\rangle \text{ subject to } \forall \boldsymbol{e} \in E : \langle \boldsymbol{e}, \boldsymbol{z}\rangle \leq 0 \text{ and } \langle \boldsymbol{c}, \boldsymbol{z}\rangle \leq 1. \tag{8.16}$$

(Here $\boldsymbol{z}$ contains the variables.) The optimum of this linear program is 0 if $\boldsymbol{x}$ is optimal and 1 otherwise. The separation problem for this polyhedron reduces to computing

$$\max_{\boldsymbol{e}\in E} \langle \boldsymbol{e}, \boldsymbol{z}\rangle.$$

In our approach to computing this maximum we need to get some control over the number of $\boldsymbol{x}$-tight tuples. To this end, we define $f' : \mathcal{M}^n \to \mathbb{Z}$ as $f'(\boldsymbol{t}) = (n^2+1) \cdot f(\boldsymbol{t}) + \rho(\boldsymbol{t})(2n - \rho(\boldsymbol{t}))$. It is not hard to see that a minimiser of f' is also a minimiser of f. Furthermore, by Lemma 8.21, f' is strictly submodular. When minimising submodular functions we can thus assume that the function is strictly submodular. By Lemma 8.22 if $\boldsymbol{x}$ is a vertex of $P_M(f')$, then T (the $\boldsymbol{x}$-tight tuples) is a chain. This implies that $|T| \leq 2n$.

Lemma 8.27. *If $f : \mathcal{M}^n \to \mathbb{Z}$ is strictly submodular and $\boldsymbol{x}$ is a vertex of $P_M(f)$, then the linear program* (8.16) *can be solved in time polynomial in n, $\log \max(|f|)$ and the encoding length of $\boldsymbol{c}$. (Assuming that the set of all $\boldsymbol{x}$-tight tuples T is available to the algorithm.)*

Proof. As f is strictly submodular it follows from Lemma 8.22 that $|T| \leq 2n$. Hence, the separation problem for (8.16) can be solved in polynomial time. By the equivalence of separation and optimisation given by the Ellipsoid algorithm it follows that (8.16) can be solved in time polynomial in n, $\log \max(|f|)$ and the encoding length of $\boldsymbol{c}$. (Note that even though $|T| \leq 2n$, the number of inequalities in E may be exponential in n. In particular the tuple $\mathbf{1}_{\mathcal{M}^n}$ can induce as many as $\binom{|A|}{2}^n$ inequalities.) □

By the algorithm in Lemma 8.27 we can find an optimal solution $\boldsymbol{z}$ to (8.16). We also need to find the largest $\delta > 0$ such that $\boldsymbol{x} + \delta \boldsymbol{z} \in P_M(f)$. We construct an algorithm for this in Lemma 8.29, but first we need an auxiliary lemma.

Lemma 8.28. *Let $\boldsymbol{y}$ be an optimal solution to* (8.16) *which is a vertex such that $\langle \boldsymbol{c}, \boldsymbol{y} \rangle = 1$. Assume that there are $\boldsymbol{t_1}, \boldsymbol{t_2} \in \mathcal{M}^n$, $\boldsymbol{t_1} \sqsubset \boldsymbol{t_2}$, and $\boldsymbol{e_1} \in E \cap I(\boldsymbol{t_1}), \boldsymbol{e_2} \in E \cap I(\boldsymbol{t_2})$ such that $\langle \boldsymbol{e_1}, \boldsymbol{y} \rangle = f(\boldsymbol{t_1})$ and $\langle \boldsymbol{e_2}, \boldsymbol{y} \rangle = f(\boldsymbol{t_2})$. Furthermore, assume that there is no $\boldsymbol{u} \in T$ such that $\boldsymbol{t_1} \sqsubset \boldsymbol{u} \sqsubset \boldsymbol{t_2}$ with any $\boldsymbol{e} \in E \cap I(\boldsymbol{u})$ and $\langle \boldsymbol{e}, \boldsymbol{y} \rangle = f(\boldsymbol{u})$. Then, there are no three distinct coordinates $i, j, k \in [n]$ such that $\boldsymbol{t_1}(i) = \boldsymbol{t_1}(j) = \boldsymbol{t_1}(k) = 0_{\mathcal{M}}$ and $\boldsymbol{t_2}(i) = \boldsymbol{t_2}(j) = \boldsymbol{t_2}(k) = 1_{\mathcal{M}}$.*

Proof. Let $E' \subseteq E$ be the vectors which define tight inequalities for $\boldsymbol{y}$. As $\boldsymbol{y}$ is a vertex and $\langle \boldsymbol{c}, \boldsymbol{y} \rangle = 1$, it follows that the polyhedron $P = \{\boldsymbol{z} \in \mathbb{R}^{[n] \times A} \mid \langle \boldsymbol{e}, \boldsymbol{z} \rangle = f(\boldsymbol{t}), \boldsymbol{e} \in E', \boldsymbol{e} \in I(\boldsymbol{t})\}$ is one dimensional.

Let $\alpha, \beta \in \mathbb{R}$ be arbitrary and define $\boldsymbol{y'} \in \mathbb{R}^{[n] \times A}$ by

$$\boldsymbol{y'} = \boldsymbol{y} + (\alpha + \beta)\boldsymbol{\chi_i} - \alpha \boldsymbol{\chi_j} - \beta \boldsymbol{\chi_k}.$$

From the non-existence of any $\boldsymbol{u} \in \mathcal{M}^n$ such that $\boldsymbol{t_1} \sqsubset \boldsymbol{u} \sqsubset \boldsymbol{t_2}$ and $\boldsymbol{e} \in E \cap I(\boldsymbol{u})$, $\langle \boldsymbol{e}, \boldsymbol{y} \rangle = f(\boldsymbol{u})$ it follows that $\langle \boldsymbol{e}, \boldsymbol{y'} \rangle = f(\boldsymbol{t})$ for all $\boldsymbol{e} \in E', \boldsymbol{e} \in I(\boldsymbol{t})$. However, this means that $\boldsymbol{y'} \in P$ and as α and β were arbitrary it follows that P is not one-dimensional. This is a contradiction and the lemma follows. □

The following lemma is a crucial part of our pseudopolynomial-time algorithm for SFM($\mathcal{M}$). With the algorithm in this lemma we are able to go from one vertex in $P_M(f)$ to a better one (if there is a better one).

Lemma 8.29. *Let $f : \mathcal{M}^n \to \mathbb{Z}$ be a strictly submodular function. Given $\boldsymbol{c} \in \mathbb{Q}^{[n]\times A}$, a vertex $\boldsymbol{x}$ of $P_M(f)$, and the set of $\boldsymbol{x}$-tight tuples T, there is an algorithm which is polynomial in n, $\log\max(|f|)$ and the encoding length of $\boldsymbol{c}$ which finds a vertex $\boldsymbol{y} \in P_M(f)$ such that $\langle \boldsymbol{c}, \boldsymbol{y}\rangle > \langle \boldsymbol{c}, \boldsymbol{x}\rangle$ or concludes that no such vertex exist. If $\boldsymbol{y}$ exists the set of $\boldsymbol{y}$-tight tuples can be computed within the same time bound.*

Proof. If there is such a vertex $\boldsymbol{y}$, then the value of the optimum of the linear program (8.16) is 1. By Lemma 8.27 this optimum $\boldsymbol{y'}$ can be found in polynomial time. The set of tuples $T' \subseteq T$ which are $\boldsymbol{y'}$-tight can be found in polynomial time (as $|T| \leq 2n$). Furthermore, by Lemma 8.28 the gap between two successive tuples in T' is not too large. It follows from Lemma 8.20 that we can find a vertex $\boldsymbol{y}$ of $P_M(f)$ such that $\langle \boldsymbol{c}, \boldsymbol{y}\rangle > \langle \boldsymbol{c}, \boldsymbol{x}\rangle$ in polynomial time.

It remains to find the rest of the $\boldsymbol{y'}$-tight tuples within the stated time bound. By Lemma 8.28 for any consecutive tuples $\boldsymbol{a}, \boldsymbol{b}$ in T' there are at most two distinct coordinates $i, j \in [n]$ such that $\boldsymbol{a}(i) = \boldsymbol{a}(j) = 0_{\mathcal{M}}$ and $\boldsymbol{b}(i) = \boldsymbol{b}(j) = 1_{\mathcal{M}}$. We will show that for every such pair $\boldsymbol{a}, \boldsymbol{b}$ in T' we can find the $\boldsymbol{y}$-tight tuples $\boldsymbol{t}$ which satisfies $\boldsymbol{a} \sqsubset \boldsymbol{t} \sqsubset \boldsymbol{b}$. To do this, for each $p, q \in M$, we find the minimisers to the submodular function $f_{p,q}$ defined as $f_{p,q}(\boldsymbol{x}) = f(\boldsymbol{x}[i = p, j = q]) - \boldsymbol{y}(\boldsymbol{x}[i = p, j = q])$ over the set $X = \{\boldsymbol{x} \in M^n \mid \boldsymbol{a} \sqsubset \boldsymbol{x} \sqsubset \boldsymbol{b}\}$. As f is submodular and $\boldsymbol{y}$ is supermodular it follows that f' is submodular. To minimise $f_{p,q}$ over X we can minimise it over at most $n^2|A|^2$ intervals defined by $\{\boldsymbol{x} \in M^n \mid \boldsymbol{a^*} \sqsubseteq \boldsymbol{x} \sqsubseteq \boldsymbol{b_*}\}$ where $\boldsymbol{a} \prec \boldsymbol{a^*}$ and $\boldsymbol{b_*} \prec \boldsymbol{b}$ (there are at most $n|A|$ choices for $\boldsymbol{a^*}$ and at most $n|A|$ choices for $\boldsymbol{b_*}$).

Note that each of these intervals is a product of the two element lattice and hence this minimisation can be done with the known algorithms for minimising submodular set functions. We can use this method to find all minimisers of $f_{p,q}$ in the interval we are interested in. (When we have found one minimiser $\boldsymbol{m}$ we iteratively minimise $f_{p,q}$ over the sets $\{\boldsymbol{x} \in M^n \mid \boldsymbol{a} \sqsubset \boldsymbol{x} \sqsubset \boldsymbol{m}\}$ and $\{\boldsymbol{x} \in M^n \mid \boldsymbol{m} \sqsubset \boldsymbol{x} \sqsubset \boldsymbol{b}\}$.) As the $\boldsymbol{y}$-tight tuples is a chain in M^n there are only a polynomial number of $\boldsymbol{y}$-tight tuples and hence this step of the algorithm runs in polynomial time. Hence the set of all $\boldsymbol{y}$-tight tuples can be found within the stated time bound. □

We are now finally ready to show the existence of a pseudopolynomial-time separation algorithm for $P_M(f)$.

Theorem 8.30. *Let $f : \mathcal{M}^n \to \mathbb{Z}$ be submodular. It is possible to decide if $\mathbf{0}$ is contained in $P_M(f)$ or not in time polynomial in n and $\max(|f|)$.*

Proof. By the equivalence of separation and optimisation given by the Ellipsoid algorithm there is an algorithm which decides if $\mathbf{0}$ is contained in $P_M(f)$ or not which makes use of an optimisation oracle for $P_M(f)$. The number of calls to the optimisation oracle is bounded by a polynomial in n and $\log\max(|f|)$, furthermore the objective function given to the optimisa-

tion oracle is given by a vector $\boldsymbol{c} \in \mathbb{Q}^{[n]\times A}$ such that the encoding length of $\boldsymbol{c}$ is bounded by a polynomial in n and $\log \max(|f|)$.

To prove the lemma it is therefore sufficient to construct an algorithm such that given $\boldsymbol{c} \in \mathbb{Z}^{[n]\times A}$ (there is no loss of generality in assuming that $\boldsymbol{c}$ is integral, a simple scaling of $\boldsymbol{c}$ achieves this) it solves $\max\langle \boldsymbol{y}, \boldsymbol{c}\rangle, \boldsymbol{y} \in P_M(f)$ in time polynomial in n, $\max(|f|)$ and the size of the encoding of $\boldsymbol{c}$. Let $f'(\boldsymbol{t}) = (n^2+1)\cdot f(\boldsymbol{t}) + \rho(\boldsymbol{t})(2n - \rho(\boldsymbol{t}))$. By Lemma 8.21 f' is strictly submodular. Furthermore, it is easy to see that any minimiser of f' is also a minimiser of f. By Lemma 8.22 each vertex $\boldsymbol{x}$ of $P_M(f')$ is "characterised" by a chain of $\boldsymbol{x}$-tight tuples.

The algorithm consists of a number of iterations. In iteration j a current vertex $\boldsymbol{x_j}$ of $P_M(f')$ is computed together with its associated chain $\boldsymbol{C_j}$ of $\boldsymbol{x_j}$-tight tuples. The initial vertex $\boldsymbol{x_0}$ and initial chain $\boldsymbol{C_0}$ is computed by the greedy algorithm from Lemma 8.3.

In iteration j, either $\boldsymbol{x_j}$ is the optimum or there is some other vertex $\boldsymbol{x_{j+1}}$ such that $\langle \boldsymbol{x_{j+1}}, \boldsymbol{c}\rangle > \langle \boldsymbol{x}, \boldsymbol{c}\rangle$. To find such an $\boldsymbol{x_{j+1}}$ or conclude that no such vertex exists we use the algorithm from Lemma 8.29. In the case when $\boldsymbol{x_{j+1}}$ exists we also get the chain $\boldsymbol{C_{j+1}}$ of $\boldsymbol{x_{j+1}}$-tight tuples from the algorithm in Lemma 8.29.

By Theorem 8.26 the vertices of $P_M(f)$ are half-integral. This implies that $\langle \boldsymbol{x_{j+1}}, \boldsymbol{c}\rangle \geq \langle \boldsymbol{x_j}, \boldsymbol{c}\rangle + 1/2$. So the algorithm is polynomial if we can prove that the optimum value is not too far from the starting point $\boldsymbol{x}_0$. That is, the difference between $\langle \boldsymbol{c}, \boldsymbol{x}_0\rangle$ and $\max\langle \boldsymbol{c}, \boldsymbol{y}\rangle, \boldsymbol{y} \in P_M(f)$ should be bounded by a polynomial in n, $\max(|f|)$ and the encoding length of $\boldsymbol{c}$. Note that as the size of the encoding of $\boldsymbol{c}$ is bounded by a polynomial in n and $\log \max(|f|)$ it follows that $\max_{i\in[n],a\in A} |\boldsymbol{c}(i,a)|$ is bounded by a polynomial in n and $\max(|f|)$. Furthermore, as $\boldsymbol{x}_0$ is obtained by the greedy algorithm it follows that for any $i \in [n], a \in A$ we have $-2\max(|f|) \leq \boldsymbol{x}_0(i,a)$. We now obtain the inequality

$$-2\max(|f|)\cdot n|A|\left(\max_{i\in[n],a\in A}|\boldsymbol{c}(i,a)|\right) \leq \langle \boldsymbol{c}, \boldsymbol{x}_0\rangle \leq \langle \boldsymbol{c}, \boldsymbol{y}\rangle \leq$$
$$\max(|f|)\cdot n|A|\left(\max_{i\in[n],a\in A}|\boldsymbol{c}(i,a)|\right).$$

From this inequality and the fact that $\max_{i\in[n],a\in A} |\boldsymbol{c}(i,a)|$ is bounded by a polynomial in n and $\max(|f|)$ it follows that the difference between $\langle \boldsymbol{c}, \boldsymbol{x}_0\rangle$ and $\langle \boldsymbol{c}, \boldsymbol{y}\rangle$ is bounded by a polynomial in n and $\max(|f|)$. As the objective function increases by at least $1/2$ in each iteration this implies that the number of iterations is bounded by a polynomial in n and $\max(|f|)$. □

From Theorem 8.30 we now get our desired result. The proof of this corollary was given in Section 8.7.

Corollary 8.31. *For every $k \geq 3$ the lattice $\mathcal{M}_k$ is oracle-pseudo-tractable.*

Chapter 9

SFM on Modular Lattices

9.1 Introduction

In this chapter we will generalise the min–max result obtained in Chapter 8 to modular atomistic finite lattices. We will also generalise the good characterisation results to general modular lattices (not necessarily atomistic). Recall that the atoms of a lattice $\mathcal{L}$ are the elements which cover $0_{\mathcal{L}}$. That is, $a \in \mathcal{L}$ is an atom if and only if $0_{\mathcal{L}} \prec a$. A lattice $\mathcal{L}$ is said to be *atomistic* if for every $x \in \mathcal{L}$ there is a subset X of the atoms of $\mathcal{L}$ such that

$$x = \bigsqcup_{a \in X} a.$$

With these definitions it is clear that the diamonds are atomistic and as they also are modular the results in this chapter applies to a class of lattices which contains the diamonds. Another example of modular atomistic lattices are the boolean lattices: let V be a finite set, then the boolean lattice is the subsets of V ordered by $\subseteq$. These lattices are atomistic, the atoms are the one element subsets of V. These lattices are also modular (as mentioned in Chapter 7 they are distributive and every distributive lattice is modular).

To see what a lattice which is modular and atomistic but neither a diamond nor distributive looks like let n be a positive integer and let $\mathbb{F}$ be a finite field. From n and $\mathbb{F}$ one can construct the n-dimensional vector space over $\mathbb{F}$, which we denote by $\mathbb{F}^n$. When the subspaces of $\mathbb{F}^n$ are ordered by inclusion they form a partial order which is in fact a modular atomistic lattice. The rank function for this lattice is given by the dimension of the subspaces. The atomicity follows from the fact that the one dimensional subspaces of any subspace of $\mathbb{F}^n$ spans the subspace. (The one dimensional subspaces of $\mathbb{F}^n$ are the atoms of the lattice.) If $n \geq 2$ these lattices are not distributive, furthermore if $n \geq 3$, then they are not diamonds. So if $n \geq 3$, then $\mathbb{F}^n$ is an example of a lattice which is not captured by the results in Chapter 8 (and neither by the previously known result that distributive lat-

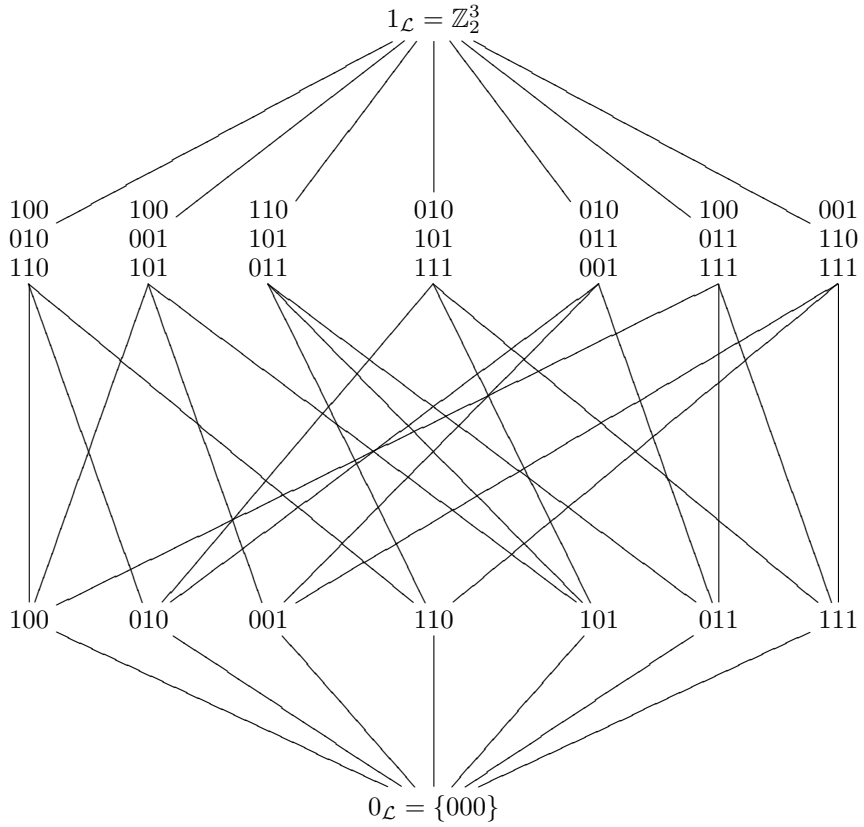


Figure 9.1: The lattice of subspaces of $\mathbb{Z}_2^3$. To make the figure less cluttered the vector 000 has been omitted from some lattice elements.

tices are oracle-tractable). As a concrete example, in Figure 9.1 the lattice of subspaces of the three dimensional vector space over the two element field is diagrammed.

In Section 9.3 we prove that there is a min–max theorem for submodular functions over direct products of modular atomistic lattices. Furthermore, in Section 9.4 we show that every modular lattice is well-characterised. These two results generalise Theorem 8.5 (min–max theorem for SFM on diamonds) and Theorem 8.19 (well-characterisedness of diamonds), respectively. We have not been able to extend Corollary 8.31 (oracle-pseudo-tractability of the diamonds) to modular atomistic lattices (and, of course, not to the more general modular lattices either). Some of the obstacles to obtaining such a result are discussed in Section 9.5.

Many of the results in this chapter have analogous results for the diamond

case in Chapter 8. The reader is strongly advised to read Chapter 8 before this chapter to get a better understanding of what is going on.

9.2 Preliminaries

Many of the definitions here have analogous definitions for the diamonds in Section 8.3. We will point out the differences and similarities as the concepts are introduced.

Let $\mathcal{L}$ be a finite modular lattice. Throughout this chapter n will denote a positive integer. We use ρ for the rank function for both $\mathcal{L}$ and $\mathcal{L}^n$. For an integer $i \in \{0, 1, \ldots, n\}$ we will use $\boldsymbol{v_i}$ to denote the tuple defined by $\boldsymbol{v_i}(j) = 1_{\mathcal{L}}$ for all $j \in [i]$ and $\boldsymbol{v_i}(j) = 0_{\mathcal{L}}$ otherwise. (So, in particular, $\boldsymbol{v_0} = \mathbf{0}_{\boldsymbol{\mathcal{L}^n}}$.)

For a vector $\boldsymbol{x} \in \mathbb{R}^{[n]\times\mathcal{L}}$ and tuple $\boldsymbol{t} \in \mathcal{L}^n$ we define

$$\boldsymbol{x}(\boldsymbol{t}) = \sum_{i=1}^{n} \boldsymbol{x}(i, \boldsymbol{t}(i)).$$

This is similar to how vectors were defined for diamonds, but not identical. For diamonds we defined a vector to be a function from $[n] \times A$ to $\mathbb{R}$ where A was the atoms of the diamond. For a vector $\boldsymbol{x} \in \mathbb{R}^{[n]\times A}$ we then defined $\boldsymbol{x}(\cdot, 1_{\mathcal{M}})$ in terms of the value of the vector on the atoms. For general modular lattices we instead define vectors to be functions from $[n] \times \mathcal{L}$ to $\mathbb{R}$. In some sense this is quite natural and we do not have to come up with any extra evaluation strategies as we had to do for $1_{\mathcal{M}}$ for diamonds.

A vector $\boldsymbol{x} \in \mathbb{R}^{[n]\times\mathcal{L}}$ is said to be *supermodular* if $\boldsymbol{u} \mapsto \boldsymbol{x}(\boldsymbol{u})$ is a supermodular function. Note that such a vector is supermodular if and only if $u \mapsto \boldsymbol{x}(i, u)$ is supermodular for each $i \in [n]$. Furthermore, it is easy to see that the set of supermodular vectors forms a polyhedron.

Let $f : \mathcal{L}^n \to \mathbb{R}$ be submodular and let

$$P(f) = \Big\{ \boldsymbol{x} \in \mathbb{R}^{[n]\times\mathcal{L}} \;\Big|\; \begin{array}{l} \boldsymbol{x} \text{ is supermodular and} \\ \forall i \in [n] : \boldsymbol{x}(i, 0_{\mathcal{L}}) = 0 \text{ and} \\ \forall \boldsymbol{t} \in \mathcal{L}^n : \boldsymbol{x}(\boldsymbol{t}) \leq f(\boldsymbol{t}) \end{array} \Big\}.$$

This definition is similar to how we defined $P_M(f)$ in the diamond case. The main difference is that we now require explicitly that the vectors are supermodular. In the diamond case the supermodularity of the vectors followed from how applying a vector to a tuple was defined.

The set $P(f)$ is a polyhedron as all the constraints on $\boldsymbol{x}$ are linear inequalities. As in the diamond case the number of inequalities defining $P(f)$ grows exponentially with n. Hence, we cannot use any algorithm for optimising over $P(f)$ which looks at every inequality which defines $P(f)$, if we

want to do it in time polynomial in n. We also define the related set of vectors $B(f)$ as

$$B(f) = \{\boldsymbol{x} \in P(f) \mid \boldsymbol{x}(\mathbf{1}_{\mathcal{L}^n}) = f(\mathbf{1}_{\mathcal{L}^n})\}.$$

This definition is also similar to how $B_M(f)$ was defined for diamonds. $B(f)$ can in fact be considered to be more natural as $B(f)$ is a polyhedron, something which $B_M(f)$ not necessarily was.

Let $\mathcal{L}$ be a finite modular atomistic lattice and let A be the atoms of $\mathcal{L}$. For a set X of lattice elements we will write $\sqcup X$ to denote the lattice element

$$\bigsqcup_{x \in X} x.$$

We now generalise the concept of unified vectors over diamonds (Definition 8.1) to unified vectors over modular atomistic lattices.

Definition 9.1 (Unified vector)**.** *A vector $\boldsymbol{x} \in \mathbb{R}^{\mathcal{L}}$ is* unified *if there is a chain $0_{\mathcal{L}} = c_0 \prec c_1 \prec \ldots \prec c_m = 1_{\mathcal{L}}$ in $\mathcal{L}$ such that for $i \in [m]$ if $X_i = \{a \in A \mid a \not\sqsubseteq c_{i-1}, a \sqsubseteq c_i\}$, then*

1. *for each i and $x, y \in X_i$ we have $\boldsymbol{x}(x) = \boldsymbol{x}(y)$; and*
2. *if $x \in X_i$ and $y \in X_{i+1}$, then $\boldsymbol{x}(x) \geq \boldsymbol{x}(y)$; and*
3. *$\boldsymbol{x}(0_{\mathcal{L}}) = 0$; and*
4. *for each $x \in \mathcal{L}, x \neq 0_{\mathcal{L}}$,*

$$\boldsymbol{x}(x) = \max\left\{\sum_{a \in X} \boldsymbol{x}(a) \;\middle|\; X \subseteq A, \rho(x) = |X|, \sqcup X = x\right\}. \tag{9.1}$$

We extend the definition of unified vectors to the vectors in $\mathbb{R}^{[n] \times \mathcal{L}}$ by saying that $\boldsymbol{x} \in \mathbb{R}^{[n] \times \mathcal{L}}$ is unified if $x \mapsto \boldsymbol{x}(i, x)$ is unified for each $i \in [n]$.

Given a unified vector $\boldsymbol{x} \in \mathbb{R}^{\mathcal{L}}$ we say that $(c_0, c_1, \ldots, c_m)$ and $(X_1, X_2, \ldots, X_m)$ are *associated* with $\boldsymbol{x}$.

Unified vectors enjoy two important properties: they are supermodular (Lemma 9.4) and secondly, if $\boldsymbol{x} \in \mathbb{R}^{\mathcal{L}}$ is unified and $\boldsymbol{x} \leq 0$, then $\boldsymbol{x}(1_{\mathcal{L}}) \leq \boldsymbol{x}(x)$ for all $x \in \mathcal{L}$. This latter property also holds for unified vectors in the diamond case and was used to prove Theorem 8.5 (the min–max theorem for diamonds). We will use this property in a similar way in this chapter to prove Theorem 9.7 (a min–max theorem for modular atomistic lattices).

9.3 A Min–max Theorem for Modular Atomistic Lattices

In this section we will show that a certain min–max theorem holds for submodular functions over modular atomistic lattices. This result is presented

as Theorem 9.7. We start out by proving that if a vector is unified then it is also supermodular. To this end we establish two preliminary lemmas and then, in Lemma 9.4, we get the implication we are looking for. In the diamond case this implication (every unified vector is supermodular) is, more or less, a trivial observation from the definition of a unified vector. We need considerably more work to establish the analogous result in the current setting.

Lemma 9.2. *If $\boldsymbol{x} \in \mathbb{R}^{\mathcal{L}}$ is unified and $(c_0, c_1, \ldots, c_m)$, $(X_1, X_2, \ldots, X_m)$ are associated with $\boldsymbol{x}$, then for each $x \in \mathcal{L}$ there is a subset $M \subseteq [m]$ such that*

$$\boldsymbol{x}(x) = \sum_{i \in M} \boldsymbol{x}(x_i) \quad \text{and} \quad \bigsqcup_{i \in M} x_i = x$$

where each x_i satisfies $x_i \sqsubseteq x$ but is otherwise chosen arbitrarily from X_i.

Proof. Assume, for the sake of contradiction, that there is some $x \in \mathcal{L}$ such that there are two distinct atoms a, b and $a, b \in X_i$ for some $i \in [m]$ and that $\boldsymbol{x}(a)$ and $\boldsymbol{x}(b)$ occurs in the sum (9.1).

Let C denote the chain $c_0, c_1, \ldots, c_m$. Let $y = a \sqcup b$ and let c_k be the maximal element in C such that $y \not\sqsubseteq c_k$. It follows that $c_k \not\sqsubseteq y$ or $c_{k+1} = y$, because otherwise we cannot have $c_k \prec c_{k+1}$ and $y \sqsubseteq c_{k+1}$. Furthermore, $y \sqcup c_k \in C$ (we have $y \sqsubseteq c_{k+1}$ and $c_k \prec c_{k+1}$ it follows that $y \sqcup c_k = c_{k+1}$). As ρ is modular we get

$$\rho(y \sqcap c_k) = \rho(y) + \rho(c_k) - \rho(y \sqcup c_k) = 2 + k - (k+1) = 1. \tag{9.2}$$

Hence, there is some atom $c \in A$ such that $c = y \sqcap c_k$. We continue by establishing the following claim.

Claim. $a, b \not\sqsubseteq c_k$**.**

Proof. By (9.2) c is the only atom such that $c \sqsubseteq c_k$ and $c \sqsubseteq y$, hence as a, b are distinct and $a, b \sqsubseteq y$ we cannot have $a, b \sqsubseteq c_k$. Furthermore, if one of a and b are $\sqsubseteq c_k$, then so is the other one as $a, b \in X_i$. The claim follows.

From $a, b \in X_i$ and the claim it follows that $k < i$. Hence, as $c \sqsubseteq c_k$ and $\boldsymbol{x}$ is unified, $\boldsymbol{x}(c) \geq \boldsymbol{x}(a), \boldsymbol{x}(b)$. Furthermore, $c \sqcup b = c \sqcup a = y$ (note that $a, b, c \sqsubseteq y$, $\rho(y) = 2$ and a, b, c are all distinct) so we can replace a or b by c in the sum in (9.1).

We can repeat this argument until there are no such atoms a and b. Note that $|X_1| = 1$ so we will never end up in the situation where $a, b \in X_1$ (as we have required that they are distinct). □

Lemma 9.3. *For $x \in \mathcal{L}$ let*

$$M(x) = \{i \in [m] \mid \exists y \in X_i : y \sqsubseteq x\}.$$

If $\boldsymbol{x} \in \mathbb{R}^{\mathcal{L}}$ is unified and $(c_0, c_1, \ldots, c_m)$, $(X_1, X_2, \ldots, X_m)$ are associated with $\boldsymbol{x}$, then for each $x \in \mathcal{L}$

$$\boldsymbol{x}(x) = \sum_{i \in M(x)} \boldsymbol{x}(x_i)$$

where each x_i is chosen arbitrarily from X_i.

Proof. Let M' be a subset of $[m]$ such that $|M'| = m - 1$. For any $y_1 \in X_1, y_2 \in X_2, \ldots, y_m \in X_m$ we have

$$\bigsqcup_{i \in M'} y_i \sqsubset 1_{\mathcal{L}}.$$

(As $|M'| < m$ we cannot have equality here.) Furthermore, by our choice of $X_1, \ldots, X_m$ it follows that $\sqcup_{i \in [m]} y_i = 1_{\mathcal{L}}$. Hence, if $j \notin M'$, then y_j is incomparable to $\sqcup_{i \in M'} y_i$.

By Lemma 9.2 for each $x \in \mathcal{L}$ there is $M \subseteq [m]$ and $x_1, x_2, \ldots, x_m$ such that for $i \in M$, $x_i \in X_i, x_i \sqsubseteq x$ and

$$\boldsymbol{x}(x) = \sum_{i \in M} \boldsymbol{x}(x_i) \quad \text{and} \quad \bigsqcup_{i \in M} x_i = x.$$

By the argument above, for any $x_j \in X_j$ such that $j \notin M$, then x_j is incomparable to x (indeed, the argument above shows that x_j is incomparable to x in the case when $|M| = m - 1$, which implies the other cases when $|M| < m - 1$). This means that we do not have $x_j \sqsubseteq x$. Hence, for every $j \notin M$ we do not have $\exists y \in X_j : y \sqsubseteq x$.

As the converse (for every $j \in M$ it is true that $\exists y \in X_j : y \sqsubseteq x$) follows from Lemma 9.2, the lemma follows. □

We are now ready to prove the first key lemma, that the unified vectors also are supermodular.

Lemma 9.4. *If $\boldsymbol{x} \in \mathbb{R}^{\mathcal{L}}$ is unified, then $\boldsymbol{x}$ is supermodular.*

Proof. Let $(c_0, c_1, \ldots, c_m)$, $(X_1, X_2, \ldots, X_m)$ be associated with $\boldsymbol{x}$. Let x and y be arbitrary elements in $\mathcal{L}$. By Lemma 9.3 there are subsets $M(x)$, $M(y)$, $M(x \sqcap y)$ and $M(x \sqcup y)$ of $[m]$ which corresponds $\boldsymbol{x}(x)$, $\boldsymbol{x}(y)$, $\boldsymbol{x}(x \sqcap y)$ and $\boldsymbol{x}(x \sqcup y)$, respectively. From Lemma 9.2 and Lemma 9.3 we get that the cardinalities of $M(x)$, $M(y)$, $M(x \sqcap y)$, and $M(x \sqcup y)$ are $\rho(x)$, $\rho(y)$, $\rho(x \sqcap y)$ and $\rho(x \sqcup y)$, respectively.[1]

In particular

$$|M(x)| + |M(y)| = |M(x \sqcap y)| + |M(x \sqcup y)|. \tag{9.3}$$

From Lemma 9.3 it follows that

1. if $k \in M(x)$ and $k \notin M(y)$, then $k \in M(x \sqcup y)$ and $k \notin M(x \sqcap y)$; and
2. if $k \in M(x), M(y)$ and there is some atom $a \in X_k$ such that $a \sqsubseteq x, y$, then $k \in M(x \sqcup y), M(x \sqcap y)$; and

[1] In Lemma 9.2 we start with a set X of atoms such that $|X| = \rho(x)$ and $\boldsymbol{x}(x) = \sum_{a \in X} \boldsymbol{x}(a)$. This set is then used to construct a set X' such that $\boldsymbol{x}(x) = \sum_{a \in X'} \boldsymbol{x}(a)$ and $|X'| = |X| = \rho(x)$. In Lemma 9.3 it is shown that X' corresponds to $M(x)$.

3. if $k \in M(x), M(y)$ and there is no atom $a \in X_k$ such that $a \sqsubseteq x, y$, then $k \in M(x \sqcup y)$ and $k \notin M(x \sqcap y)$.

   In this case there are distinct atoms $a, a' \in X_k$ such that $a \sqsubseteq x$ and $a' \sqsubseteq y$. By the argument used in Lemma 9.2 it follows that there is some $k' < k$ such that $k' \notin M(x), M(y)$ and some atom $a'' \in X_{k'}, a'' \sqsubseteq a \sqcup a' \sqsubseteq x \sqcup y$. Hence, $k' \in M(x \sqcup y)$. Recall that by Definition 9.1 we have $\boldsymbol{x}(x_{k'}) \geq \boldsymbol{x}(x_k)$ for any $x_{k'} \in X_{k'}$ and $x_k \in X_k$.

From (9.3) and 1–3 above we now get

$$\boldsymbol{x}(x) + \boldsymbol{x}(y) \leq \boldsymbol{x}(x \sqcup y) + \boldsymbol{x}(x \sqcap y).$$

□

In the following two lemmas we show that the greedy algorithm can be used to generate vectors which are contained in $P(f)$. In the first lemma we show that there is a unified vector $\boldsymbol{x}$ which satisfies $\boldsymbol{x}(\boldsymbol{t}) = f(\boldsymbol{t})$ for all $\boldsymbol{t}$ in a certain chain in $\mathcal{L}^n$. These lemmas corresponds to Lemma 8.3 and Lemma 8.4 in the diamond case, respectively.

Lemma 9.5. *Let $f : \mathcal{L}^n \to \mathbb{R}$ be a submodular function. There is a vector $\boldsymbol{x} \in \mathbb{R}^{[n]\times\mathcal{L}}$ such that*

- *$\boldsymbol{x}$ is unified; and*
- *$\boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c_{i,j}]) = f(\boldsymbol{v_{i-1}}[i = c_{i,j}])$ for all $i \in [n]$ and $j \in [m]$, where for $i \in [n]$, $0_{\mathcal{L}} = c_{i,0} \prec c_{i,1} \prec \ldots \prec c_{i,m} = 1_{\mathcal{L}}$ is the chain in Definition 9.1 for the vector $x \mapsto \boldsymbol{x}(i, x)$.*

Proof. Given a submodular $f : \mathcal{L}^n \to \mathbb{R}$ we will construct a vector $\boldsymbol{x}$ which satisfies the requirements in the lemma. To do this we define a sequence of atoms $a_{i,j}$ for $i \in [n]$ and $j \in [m]$. The atoms will be defined inductively and are ordered by the lexicographical order of the pairs (i, j) which are associated with them (so $(i, j) \leq (i', j')$ if and only if $i < i'$ or $(i = i'$ and $j \leq j')$). For $i \in [n]$ and $j \in \{0, 1, \ldots, m\}$ we use $c_{i,j}$ to denote

$$\bigsqcup_{k \leq j} a_{i,k}$$

(so for $i \in [n]$ we have $c_{i,0} = 0_{\mathcal{L}}$) and for $i \in [n]$ and $j \in [m]$ we use $X_{i,j}$ to denote the set $\{a \in A \mid a \not\sqsubseteq c_{i,j-1}, a \sqsubseteq c_{i,j}\}$. Note that when $a_{i',j'}$ is defined for all pairs (i', j') up to and including some (i, j), then $c_{i,j}$ and $X_{i,j}$ are defined as well.

To start the inductive definition let $a_{1,1} \in \arg\max_{a \in A} f(\boldsymbol{v_0}[1 = a])$ and set $\boldsymbol{x}(1, a_{1,1}) = f(\boldsymbol{v_0}[1 = a_{1,1}])$. For the general case, choose $a_{i,j} \in A$ so that

$$a_{i,j} \in \operatorname*{arg\,max}_{a \in A, a \not\sqsubseteq c_{i,j-1}} f(\boldsymbol{v_{i-1}}[i = c_{i,j-1} \sqcup a]).$$

For $i \in [n]$ and $j \in [m]$ set

$$\boldsymbol{x}(i, a_{i,j}) = f(\boldsymbol{v_{i-1}}[i = c_{i,j}]) - f(\boldsymbol{v_{i-1}}[i = c_{i,j-1}]). \tag{9.4}$$

We now extend the definition of $\boldsymbol{x}$ to all atoms: for $a \in A$ and $i \in [n]$ let $\boldsymbol{x}(i, a) = \boldsymbol{x}(i, a_{i,j})$ where $j \in [m]$ is chosen so that $a \in X_{i,j}$. We continue by extending the definition to $[n] \times \mathcal{L}$ as follows: for $i \in [n]$ and $x \in \mathcal{L} \backslash (A \cup \{0_{\mathcal{L}}\})$ set

$$\boldsymbol{x}(i, x) = \max \left\{ \sum_{a \in X} \boldsymbol{x}(i, a) \;\middle|\; X \subseteq A, \rho(x) = |X|, \sqcup X = x \right\}. \tag{9.5}$$

We now proceed by establishing two claims.

Claim A. For each $i \in [n]$ and $j \in \{2, 3, \ldots, m\}$ if $a \in X_{i,j-1}$ and $b \in X_{i,j}$, then $\boldsymbol{x}(i, b) \leq \boldsymbol{x}(i, a)$.

Proof. As $a \in X_{i,j-1}$ and $b \in X_{i,j}$ it follows that $a \sqcup b \sqcup c_{i,j-2} = c_{i,j}$ and by the modularity of $\mathcal{L}$ we have $(a \sqcup c_{i,j-2}) \sqcap (b \sqcup c_{i,j-2}) = c_{i,j-2}$. We now get

$$\begin{aligned} &f(\boldsymbol{v_{i-1}}[i = c_{i,j}]) + f(\boldsymbol{v_{i-1}}[i = c_{i,j-2}]) && \leq \\ &f(\boldsymbol{v_{i-1}}[i = a \sqcup c_{i,j-2}]) + f(\boldsymbol{v_{i-1}}[i = b \sqcup c_{i,j-2}]) && \leq \\ &2f(\boldsymbol{v_{i-1}}[i = c_{i,j-1}]) \end{aligned}$$

where the first inequality holds due to the submodularity of f and the second inequality follows from our choice of $a_{i,j-1}$. This is equivalent to

$$\begin{aligned} \boldsymbol{x}(i, b) &= f(\boldsymbol{v_{i-1}}[i = c_{i,j}]) - f(\boldsymbol{v_{i-1}}[i = c_{i,j-1}]) \\ &\leq f(\boldsymbol{v_{i-1}}[i = c_{i,j-1}]) - f(\boldsymbol{v_{i-1}}[i = c_{i,j-2}]) \\ &= \boldsymbol{x}(i, a) \end{aligned}$$

which is what we wanted to prove.

Claim B. $\boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c_{i,j}]) = f(\boldsymbol{v_{i-1}}[i = c_{i,j}])$ for all $i \in [n]$ and $j \in [m]$.

Proof. We prove this claim by induction over the pairs (i, j) ordered lexicographically. With the pair (i, j) we associate the tuple $\boldsymbol{v_{i-1}}[i = c_{i,j}]$. Note that $(i, j) \leq (i', j')$ if and only if $\boldsymbol{v_{i-1}}[i = c_{i,j}] \sqsubseteq \boldsymbol{v_{i'-1}}[i' = c_{i',j'}]$. As $a_{1,1} \in \arg\max_{a \in A} f(\boldsymbol{v_0}[1 = a])$ and $\boldsymbol{x}(1, a_{1,1}) = f(\boldsymbol{v_0}[1 = a_{1,1}])$ the claim holds for $(i, j) = (1, 1)$. Now assume that it holds for all pairs (i', j') such that $(i', j') \leq (i, j)$. We will prove that it holds for the pair which succeeds (i, j) in the order. If $j < m$ then the next pair is $(i, j + 1)$ and we get

$$\begin{aligned} \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c_{i,j+1}]) &= \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c_{i,j}]) + \boldsymbol{x}(i, a_{i,j+1}) \\ &= f(\boldsymbol{v_{i-1}}[i = c_{i,j}]) + f(\boldsymbol{v_{i-1}}[i = c_{i,j+1}]) \\ &\quad - f(\boldsymbol{v_{i-1}}[i = c_{i,j}]) \\ &= f(\boldsymbol{v_{i-1}}[i = c_{i,j+1}]). \end{aligned}$$

Here the first inequality follows from the definition of $\boldsymbol{x}(\cdot)$, Claim A, and (9.5). The second equality follows from the induction hypothesis and (9.4). If $j = m$ the next pair is $(i+1, 1)$ and we get

$$\begin{aligned}\boldsymbol{x}(\boldsymbol{v_i}[i+1 = c_{i+1,1}]) &= \boldsymbol{x}(\boldsymbol{v_i}) + \boldsymbol{x}(i+1, a_{i+1,1}) \\ &= f(\boldsymbol{v_i}) + f(\boldsymbol{v_i}[i+1 = c_{i+1,1}]) - f(\boldsymbol{v_i}) \\ &= f(\boldsymbol{v_i}[i+1 = c_{i+1,1}])\end{aligned}$$

The first equality follows from the definition of $\boldsymbol{x}(\cdot)$. The second equality follows from the induction hypothesis and (9.4).

By Claim A, the definitions of $a_{i,j}$, $X_{i,j}$, $c_{i,j}$, and the fact that $a \mapsto \boldsymbol{x}(i,a)$ is constant on the $X_{i,j}$:s it follows that $\boldsymbol{x}$ is unified. By Claim B $\boldsymbol{x}$ satisfies the second condition in the lemma. □

The following lemma is analogous to Lemma 8.4 in the diamond case.

Lemma 9.6. *Let $f : \mathcal{L}^n \to \mathbb{R}$ be a submodular function such that $f(\mathbf{0}_{\boldsymbol{\mathcal{L}^n}}) \geq 0$. If $\boldsymbol{x} \in \mathbb{R}^{[n]\times\mathcal{L}}$ is supermodular and for each $i \in [n]$ there is a chain $0_{\mathcal{L}} \prec c_{i,1} \prec \ldots \prec c_{i,m} = 1_{\mathcal{L}}$ such that for all $i \in [n]$ and $j \in [m]$ we have $\boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c_{i,j}]) = f(\boldsymbol{v_{i-1}}[i = c_{i,j}])$, then $\boldsymbol{x} \in B(f)$.*

Proof. We will prove by induction that $\boldsymbol{x}(\boldsymbol{y}) \leq f(\boldsymbol{y})$ for all $\boldsymbol{y} \in \mathcal{L}^n$. The induction will be over the pairs $[n]\times[m]$ ordered lexicographically (so $(i,j) \leq (i',j')$ if and only if $i < i'$ or ($i = i'$ and $j \leq j'$)). With the pair (i,j) we associate the tuples $\boldsymbol{y} \in \mathcal{L}^n$ such that $\boldsymbol{y} \sqsubseteq \boldsymbol{v_{i-1}}[i = c_{i,j}]$. As

$$\boldsymbol{x}(\boldsymbol{v_0}) = \boldsymbol{x}(\mathbf{0}_{\boldsymbol{\mathcal{L}^n}}) = 0 \text{ and } f(\mathbf{0}_{\boldsymbol{\mathcal{L}^n}}) \geq 0$$

and

$$\boldsymbol{x}(\boldsymbol{v_0}[1 = c_{1,1}]) = f(\boldsymbol{v_0}[1 = c_{1,1}])$$

the statement holds for the pair $(1,1)$ (which corresponds to $\boldsymbol{y} \sqsubseteq \mathbf{0}_{\boldsymbol{\mathcal{L}^n}}[1 = c_{1,1}] = \boldsymbol{v_0}[1 = c_{1,1}]$). Let $i \in [n], j \in [m]$ and $\boldsymbol{y} \in \mathcal{L}^n, \boldsymbol{y} \sqsubseteq \boldsymbol{v_{i-1}}[i = c_{i,j}]$ and assume that the inequality holds for all $\boldsymbol{y'} \in \mathcal{L}^n$ such that $\boldsymbol{y'} \sqsubseteq \boldsymbol{v_{i'-1}}[i' = c_{i',j'}]$ where (i',j') is the predecessor to the pair (i,j). We will prove that the inequality holds for all $\boldsymbol{y} \sqsubseteq \boldsymbol{v_{i-1}}[i = c_{i,j}]$.

Case A: $\boldsymbol{i \neq i'}$**.** In this case $i = i' + 1$, $j = 1$, and $j' = m$. Hence, we can assume that $\boldsymbol{y}(i) = c_{i,1}$. We now get

$$\begin{aligned}\boldsymbol{x}(\boldsymbol{y}) &\leq \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c_{i,1}]) - \boldsymbol{x}(\boldsymbol{v_{i-1}}) + \boldsymbol{x}(\boldsymbol{y}[i = 0_{\mathcal{L}}]) \\ &\leq \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c_{i,1}]) - \boldsymbol{x}(\boldsymbol{v_{i-1}}) + f(\boldsymbol{y}[i = 0_{\mathcal{L}}]) \\ &\leq f(\boldsymbol{v_{i-1}}[i = c_{i,1}]) - f(\boldsymbol{v_{i-1}}) + f(\boldsymbol{y}[i = 0_{\mathcal{L}}]) \\ &\leq f(\boldsymbol{y}).\end{aligned}$$

The first inequality follows from the supermodularity of $\boldsymbol{x}$. The second inequality follows from the induction hypothesis and the fact that $\boldsymbol{y}[i = 0_{\mathcal{L}}] \sqsubseteq \boldsymbol{v_{i-1}}$. The third inequality follows from the assumptions in the statement

of the lemma. Finally, the last inequality follows from the submodularity of f.

Case B: $i = i'$**.** In this case $j' < m$ and $j = j' + 1$. To simplify the notation a bit we let $y = \boldsymbol{y}(i)$. We will also use C_i to denote the chain $c_{i,0}, c_{i,1}, \ldots, c_{i,m}$ where we let $c_{i,0} = 0_{\mathcal{L}}$.

We can assume that $c_{i,j-1}$ is the maximal element in C_i such that $y \not\sqsubseteq c_{i,j-1}$ (otherwise we would get $\boldsymbol{x}(\boldsymbol{y}) \leq f(\boldsymbol{y})$ from the induction hypothesis). It follows that $c_{i,j-1} \not\sqsubseteq y$ or $y = c_{i,j}$, because otherwise we cannot have $c_{i,j-1} \prec c_{i,j}$ and $y \sqsubseteq c_{i,j}$. Furthermore, $y \sqcup c_{i,j-1} \in C_i$ (in fact $y \sqcup c_{i,j-1} = c_{i,j}$). Now,

$$\begin{aligned}
\boldsymbol{x}(\boldsymbol{y}) &\leq \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = y \sqcup c_{i,j-1}]) - \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = c_{i,j-1}]) + \boldsymbol{x}(\boldsymbol{y}[i = y \sqcap c_{i,j-1}]) \\
&\leq \boldsymbol{x}(\boldsymbol{v_{i-1}}[i = y \sqcup c_{i,j-1}]) - \boldsymbol{x}(\boldsymbol{v_{i-1}}[l = c_{i,j-1}]) + f(\boldsymbol{y}[i = y \sqcap c_{i,j-1}]) \\
&\leq f(\boldsymbol{v_{i-1}}[i = y \sqcup c_{i,j-1}]) - f(\boldsymbol{v_{i-1}}[i = c_{i,j-1}]) + f(\boldsymbol{y}[i = y \sqcap c_{i,j-1}]) \\
&\leq f(\boldsymbol{y}).
\end{aligned}$$

The first inequality follows from the supermodularity of $\boldsymbol{x}$. The second inequality follows from the induction hypothesis and the fact that $y \sqcap c_{i,j-1} \sqsubseteq c_{i,j-1}$. The third inequality follows from $y \sqcup c_{i,j-1}, c_{i,j-1} \in C_i$ and the assumptions in the statement of the lemma. Finally, the last inequality follows from the submodularity of f.

We have shown that $\boldsymbol{x} \in P(f)$. By the assumption in the lemma $\boldsymbol{x}(\boldsymbol{v_{n-1}}[n = c_{n,m}]) = f(\boldsymbol{v_{n-1}}[n = c_{n,m}])$ and $\boldsymbol{v_{n-1}}[n = c_{n,m}] = \mathbf{1}_{\boldsymbol{\mathcal{L}^n}}$ so $\boldsymbol{x} \in B(f)$. □

We are now ready to state and prove the min–max theorem. Both the statement and proof of this theorem is similar to the analogous result for the diamonds, Theorem 8.5.

Theorem 9.7. *Let $f : \mathcal{L}^n \to \mathbb{R}$ be a submodular function. If $f(\mathbf{0}_{\boldsymbol{\mathcal{L}^n}}) = 0$, then*

$$\min_{\boldsymbol{x} \in \mathcal{L}^n} f(\boldsymbol{x}) = \max \left\{ \boldsymbol{z}(\mathbf{1}_{\boldsymbol{\mathcal{L}^n}}) \mid \boldsymbol{z} \in P(f), \boldsymbol{z} \leq 0, \boldsymbol{z} \text{ is unified} \right\}.$$

Moreover, if f is integer-valued then there is an integer-valued vector $\boldsymbol{z}$ which maximises the right hand side.

Proof. If $\boldsymbol{z} \in P(f)$, $\boldsymbol{z} \leq 0$, and $\boldsymbol{z}$ is unified then, by Lemma 9.3, we get

$$\boldsymbol{z}(\mathbf{1}_{\boldsymbol{\mathcal{L}^n}}) \leq \boldsymbol{z}(\boldsymbol{y}) \leq f(\boldsymbol{y})$$

for any $\boldsymbol{y} \in \mathcal{L}^n$. Hence, LHS $\geq$ RHS holds. Consider the function $f' : \mathcal{L}^n \to \mathbb{R}$ defined by

$$f'(\boldsymbol{x}) = \min_{\boldsymbol{y} \sqsubseteq \boldsymbol{x}} f(\boldsymbol{y}).$$

We continue by establishing three claims.

Claim A. f' **is submodular.**
Proof. Let $\boldsymbol{x'}, \boldsymbol{y'} \in \mathcal{L}^n$ and let $\boldsymbol{x} \sqsubseteq \boldsymbol{x'}, \boldsymbol{y} \sqsubseteq \boldsymbol{y'}$ be tuples such that $f'(\boldsymbol{x'}) = f(\boldsymbol{x})$ and $f'(\boldsymbol{y'}) = f(\boldsymbol{y})$. Now,

$$\begin{aligned} f'(\boldsymbol{x'}) + f'(\boldsymbol{y'}) = f(\boldsymbol{x}) + f(\boldsymbol{y}) &\geq f(\boldsymbol{x} \sqcap \boldsymbol{y}) + f(\boldsymbol{x} \sqcup \boldsymbol{y}) \\ &\geq f'(\boldsymbol{x'} \sqcap \boldsymbol{y'}) + f'(\boldsymbol{x'} \sqcup \boldsymbol{y'}) \end{aligned}$$

where the first equality follows from the definition of f', $\boldsymbol{x}$ and $\boldsymbol{y}$, the first inequality follows from the submodularity of f and the second inequality from the definition of f' and $\boldsymbol{x} \sqcap \boldsymbol{y} \sqsubseteq \boldsymbol{x'} \sqcap \boldsymbol{y'}$ and $\boldsymbol{x} \sqcup \boldsymbol{y} \sqsubseteq \boldsymbol{x'} \sqcup \boldsymbol{y'}$.

Claim B. For any $\boldsymbol{z} \in P(f')$ **we have** $\boldsymbol{z} \leq 0$**.**
Proof. As $f(\mathbf{0}_{\mathcal{L}^n}) = 0$ we have $f'(\boldsymbol{x}) \leq 0$ for any $\boldsymbol{x} \in \mathcal{L}^n$. For $i \in [n]$ and $a \in A$ define $\boldsymbol{t_{i,a}} \in \mathcal{L}^n$ such that $\boldsymbol{t_{i,a}}(j) = 0_{\mathcal{L}}$ for $j \in [n], j \neq i$ and $\boldsymbol{t_{i,a}}(i) = a$. It follows from $\boldsymbol{z} \in P(f')$ that we have $\boldsymbol{z}(\boldsymbol{t_{i,a}}) = \boldsymbol{z}(i,a) \leq f'(\boldsymbol{t_{i,a}}) \leq 0$ for any $a \in A$ and $i \in [n]$.

Claim C. Any $\boldsymbol{z} \in B(f')$ **satisfies** $\boldsymbol{z}(\mathbf{1}_{\mathcal{L}^n}) = f'(\mathbf{1}_{\mathcal{L}^n})$**.**
Proof. Follows from the definition of $B(f')$.

Finally, $f'(\mathbf{1}_{\mathcal{L}^n}) = \min_{\boldsymbol{x} \in \mathcal{L}^n} f(\boldsymbol{x})$ which follows from the definition of f'. From Lemma 9.5, Lemma 9.4, and Lemma 9.6 it follows that there is a unified vector $\boldsymbol{z}$ in $B(f') \subseteq P(f') \subseteq P(f)$. By Claim B $\boldsymbol{z} \leq 0$ and by Claim C $\boldsymbol{z}(\mathbf{1}_{\mathcal{L}^n}) = f'(\mathbf{1}_{\mathcal{L}^n})$. Hence, LHS $\leq$ RHS holds. To prove the existence of an integer-valued vector, note that the vector constructed in Lemma 9.5 is integer-valued if f' is integer-valued and f' is integer-valued if f is integer-valued. □

9.4 A Good Characterisation of Modular Lattices

9.4.1 Introduction

In this section we show that all modular finite lattices (not only the atomistic ones) are well-characterised. Let $\mathcal{L}$ be an arbitrary finite modular lattice and let $f : \mathcal{L}^n \to \mathbb{Z}$ be a submodular function. Instead of starting with the min–max theorem (which we did for the diamonds) we will use a slightly different approach, as we do not have a min–max theorem for general modular lattices. The fundamental observation we will use is that $\mathbf{0} \in P(f - m)$ if and only if $\min_{\boldsymbol{t} \in \mathcal{L}^n} f(\boldsymbol{t}) \geq m$. So if we are provided with a tuple $\boldsymbol{m}$, an integer $m \in \mathbb{Z}$ such that $f(\boldsymbol{m}) = m$, and a proof, which can be verified in time polynomial in n, that $\mathbf{0} \in P(f - m)$, then we can conclude that $\min_{\boldsymbol{t} \in \mathcal{L}^n} f(\boldsymbol{t}) = m$.

To use this observation we need a way to decide if $\mathbf{0} \in P(f')$ for a submodular function $f' = f - m$. To do this we use the same techniques as in the diamond case: use Carathéodory's to reduce the problem to the

vertices of $P(f')$ and prove that there are chains of tight tuples of any vertex of $P(f')$ which is "dense" in a certain specific sense. By an induction on the number of elements of the lattice (that is, on $|\mathcal{L}|$) the result is obtained.

9.4.2 Proofs

The main result of this section, that all modular lattices are well-characterised, is stated as Theorem 9.14. Most of the lemmas in this section have an analogous lemma in Section 8.6, where we proved that the diamonds are well-characterised. Some of the proofs are more or less identical to the diamond case and have therefore been omitted.

We first need some initial observations about $P(f)$. It is not hard to see that $P(f)$ is *pointed*, that is, there are vertices in $P(f)$. Indeed, $P(f)$ is pointed if and only if the lineality space of $P(f)$ is zero dimensional. [132, Chapter 8] However, for every $\boldsymbol{x} \in P(f)$, $i \in [n]$ and $x \in \mathcal{L}$ we have $\boldsymbol{x}(i, x) \leq f(\mathbf{0}_{\mathcal{L}^n}[i = x])$ and hence the lineality space of $P(f)$ is zero dimensional. We will need the following lemma.

Lemma 9.8. *Let $f : \mathcal{L}^n \to \mathbb{R}$ be a submodular function. If there is some vector $\boldsymbol{x} \in P(f)$ such that $\mathbf{0} \leq \boldsymbol{x}$, then $\mathbf{0} \in P(f)$.*

Proof. As $\mathbf{0} \leq \boldsymbol{x}$ we have $0 \leq \boldsymbol{x}(\boldsymbol{t})$ for every $\boldsymbol{t} \in \mathcal{L}^n$. It is clear that the vector $\mathbf{0}$ is supermodular. Hence, $0 = \mathbf{0}(\boldsymbol{t}) \leq \boldsymbol{x}(\boldsymbol{t}) \leq f(\boldsymbol{t})$ for every $\boldsymbol{t} \in \mathcal{L}^n$ and thus $\mathbf{0} \in P(f)$. □

The following lemma is the analogue to Lemma 8.15 for the diamond case. The proof is essentially the same, except that we have an additional assumption in the statement of the lemma, namely that all lattices with a smaller domain are well-characterised. We will use this lemma as Lemma 8.15 was used in Section 8.6 together with an induction step to deal with larger and larger lattices. We also need to use Lemma 7.6 to deal with a direct product of, possibly different, proper sublattices of $\mathcal{L}$.

Lemma 9.9. *Let $\mathcal{L}$ be a modular lattice and assume that all modular lattices $\mathcal{L}'$ such that $|\mathcal{L}'| < |\mathcal{L}|$ are well-characterised. Let n be a positive integer and let $f : \mathcal{L}^n \to \mathbb{R}$ be a submodular function which is provided to us by a value-giving oracle. Let $\boldsymbol{x} \in \mathbb{R}^{[n] \times \mathcal{L}}$ and $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{L}^n$ such that $\boldsymbol{a} \sqsubseteq \boldsymbol{b}$, $\boldsymbol{a}$ is $\boldsymbol{x}$-tight, and there are at most k coordinates $i \in [n]$ such that $\boldsymbol{a}(i) = 0_{\mathcal{L}}$ and $\boldsymbol{b}(i) = 1_{\mathcal{L}}$. Under the assumption that for all $\boldsymbol{t} \sqsubseteq \boldsymbol{a}$ we have $\boldsymbol{x}(\boldsymbol{t}) \leq f(\boldsymbol{t})$, if the statement $\forall \boldsymbol{y} \in \mathcal{L}^n : \boldsymbol{y} \sqsubseteq \boldsymbol{b} \Rightarrow \boldsymbol{x}(\boldsymbol{y}) \leq f(\boldsymbol{y})$ is true, then there is a proof of this fact which can be checked in time $O\left(|\mathcal{L}|^k \cdot n^c\right)$, for some constant c which depends on $\mathcal{L}$ but not on n or k.*

The idea in the proof of this lemma is essentially the same as the idea in Lemma 8.15: we verify that the inequality holds for all $\boldsymbol{y} \in \mathcal{L}^n$ such that $\boldsymbol{a} \sqsubseteq \boldsymbol{y} \sqsubseteq \boldsymbol{b}$ by verifying that the minimum value of a certain submodular function, defined on some product of lattices $\mathcal{L}'$ which satisfies $|\mathcal{L}'| < |\mathcal{L}|$, is greater than zero. By the assumption in the lemma there are proofs of such

statements which can be verified in polynomial time. We then show that this is sufficient to verify the inequality for all $\boldsymbol{y} \in \mathcal{L}^n$ such that $\boldsymbol{y} \sqsubseteq \boldsymbol{b}$.

Proof. Let $I \subseteq [n]$ be the set of coordinates such that $\boldsymbol{a}(i) = 0_{\mathcal{L}}$ and $\boldsymbol{b}(i) = 1_{\mathcal{L}}$. Let $m = n - |I|$ and let $p : [n] \setminus I \to \{1, 2, \ldots, m\}$ be the order preserving bijection from $[n] \setminus I$ to $\{1, 2, \ldots, m\}$ (so $p(x) < p(x')$ whenever $x < x'$). For $j \in [m]$ let $\mathcal{J}_i$ be the interval $\{x \mid \boldsymbol{a}(p^{-1}(j)) \sqsubseteq x \sqsubseteq \boldsymbol{b}(p^{-1}(j))\}$ in $\mathcal{L}$. Let $\mathcal{J}$ be the lattice

$$\mathcal{J}_1 \times \mathcal{J}_2 \times \ldots \times \mathcal{J}_m.$$

As $\mathcal{L}$ is modular the lattices $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_m$ are modular as well. For a tuple $\boldsymbol{y} \in \mathcal{J}$ we will write $p^{-1}(\boldsymbol{y})$ to denote the tuple

$$\mathbf{0}_{\mathcal{L}^n}[p^{-1}(1) = \boldsymbol{y}(1), p^{-1}(2) = \boldsymbol{y}(2), \ldots, p^{-1}(m) = \boldsymbol{y}(m)].$$

Let $Z = \{\boldsymbol{z} \in \mathcal{L}^n \mid \forall i \in [n] \setminus I : \boldsymbol{z}(i) = 0_{\mathcal{L}}\}$. For a tuple $\boldsymbol{z} \in Z$ define $g_{\boldsymbol{z}} : \mathcal{J} \to \mathbb{R}$ as

$$g_{\boldsymbol{z}}(\boldsymbol{y}) = f(\boldsymbol{z} \sqcup p^{-1}(\boldsymbol{y})). \tag{9.6}$$

The idea is now to show that $g_{\boldsymbol{z}}$ is submodular for every $\boldsymbol{z} \in Z$. As $|Z| \leq |\mathcal{L}|^k$, which is constant if $|I| = k$ is constant, and $\mathcal{J}$ consists of a product of lattices with strictly fewer than $|\mathcal{L}|$ elements there are proofs of the minimum of $g_{\boldsymbol{z}}$ which can be verified in polynomial time.

To make the idea precise we first show that $g_{\boldsymbol{z}}$ is a submodular function for each $\boldsymbol{z} \in Z$. To see this, let $\boldsymbol{z} \in Z$ and let $\boldsymbol{c}, \boldsymbol{d} \in \mathcal{J}$. We now get

$$\begin{aligned} g_{\boldsymbol{z}}(\boldsymbol{c}) + g_{\boldsymbol{z}}(\boldsymbol{d}) &= f(\boldsymbol{z} \sqcup p^{-1}(\boldsymbol{c})) + f(\boldsymbol{z} \sqcup p^{-1}(\boldsymbol{d})) \\ &\geq f(\boldsymbol{z} \sqcup p^{-1}(\boldsymbol{c}) \sqcup p^{-1}(\boldsymbol{d})) + \\ &\qquad f((\boldsymbol{z} \sqcup p^{-1}(\boldsymbol{d})) \sqcap (\boldsymbol{z} \sqcup p^{-1}(\boldsymbol{c}))) \\ &= g_{\boldsymbol{z}}(\boldsymbol{c} \sqcup \boldsymbol{d}) + g_{\boldsymbol{z}}(\boldsymbol{c} \sqcap \boldsymbol{d}), \end{aligned} \tag{9.7}$$

where the inequality follows from the submodularity of f and the last equality follows from $p^{-1}(\boldsymbol{c}) \sqcup p^{-1}(\boldsymbol{d}) = p^{-1}(\boldsymbol{c} \sqcup \boldsymbol{d})$ and $p^{-1}(\boldsymbol{c}) \sqcap p^{-1}(\boldsymbol{d}) = p^{-1}(\boldsymbol{c} \sqcap \boldsymbol{d})$ and the modularity of $\mathcal{J}$. Hence $g_{\boldsymbol{z}}$ is submodular for each $\boldsymbol{z} \in Z$.

Analogous to (9.6) above, for a tuple $\boldsymbol{z} \in Z$ we define $h_{\boldsymbol{z}} : \mathcal{J} \to \mathbb{R}$ as

$$h_{\boldsymbol{z}}(\boldsymbol{y}) = \boldsymbol{x}(\boldsymbol{z} \sqcup p^{-1}(\boldsymbol{y})).$$

By using the same argument as for $g_{\boldsymbol{z}}$ one can show that $h_{\boldsymbol{z}}$ is supermodular for each $\boldsymbol{z} \in Z$.

Let $\boldsymbol{y} \in \mathcal{J}$ and let $\boldsymbol{z} \in Z$. For a fixed k the inequalities

$$\begin{aligned} &\boldsymbol{x}(\boldsymbol{z} \sqcup p^{-1}(\boldsymbol{y})) \leq \\ &f(\boldsymbol{z} \sqcup p^{-1}(\boldsymbol{y})) \\ &\quad \Longleftrightarrow \\ &0 \leq g_{\boldsymbol{z}}(\boldsymbol{y}) - h_{\boldsymbol{z}}(\boldsymbol{y}) \end{aligned} \tag{9.8}$$

can be verified to hold for every $\boldsymbol{y} \in \mathcal{J}$ and $\boldsymbol{z} \in Z$ in time $O(|\mathcal{L}|^k \cdot n^c)$ as, for each $\boldsymbol{z} \in Z$, the RHS of (9.8) is a submodular function in $\boldsymbol{y}$, over a product of lattices with strictly fewer elements than $\mathcal{L}$. To see this first assume that (9.8) does indeed hold. We have that $|\mathcal{L}_i| < |\mathcal{L}|$ for $i \in [m]$ and hence, by the assumptions in the lemma, $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_m$ are well-characterised. Hence, by Lemma 7.6 $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_m\}$ is well-characterised. This means that, for each fixed $\boldsymbol{z}$, there are proofs which can be checked in time $O(n^c)$ that the minimum over all $\boldsymbol{y} \in \mathcal{J}$ of the RHS of (9.8) is not less than 0. As $|Z| = |\mathcal{L}|^k$ we can perform this minimisation for all $\boldsymbol{z} \in Z$ in time $O(|\mathcal{L}|^k \cdot n^c)$. Conversely, if (9.8) does not hold for some $\boldsymbol{y} \in \mathcal{J}$ and $\boldsymbol{z} \in Z$, then there is a tuple $\boldsymbol{t}$ such that $\boldsymbol{a} \sqsubseteq \boldsymbol{t} \sqsubseteq \boldsymbol{b}$ and $\boldsymbol{x}(\boldsymbol{t}) \not\leq f(\boldsymbol{t})$.

One can now proceed as in Lemma 8.15 and show that if $\boldsymbol{x}(\boldsymbol{t}) \leq f(\boldsymbol{t})$ for all $\boldsymbol{t} \in \mathcal{L}^n$ such that $\boldsymbol{t} \sqsubseteq \boldsymbol{a}$ or $\boldsymbol{a} \sqsubseteq \boldsymbol{t} \sqsubseteq \boldsymbol{b}$, then $\boldsymbol{x}(\boldsymbol{t}) \leq f(\boldsymbol{t})$ for all $\boldsymbol{t} \in \mathcal{L}^n$ such that $\boldsymbol{t} \sqsubseteq \boldsymbol{b}$. We refer the reader to the proof of Theorem 8.19 for the details. □

We will need the following lemma which is analogous to Lemma 8.2 in the diamond case. The proof is the same, so we refer to the proof of Lemma 8.2 for the details.

Lemma 9.10. *Let $f : \mathcal{L}^n \to \mathbb{R}$ be a submodular function. Let $\boldsymbol{x} \in P(f)$ be a vector and let $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{L}^n$ be $\boldsymbol{x}$-tight tuples. Then, $\boldsymbol{a} \sqcup \boldsymbol{b}$ and $\boldsymbol{a} \sqcap \boldsymbol{b}$ are $\boldsymbol{x}$-tight.*

The lemma below is the translation of Lemma 8.14 to modular lattices.

Lemma 9.11. *Let $f : \mathcal{L}^n \to \mathbb{R}$ be a submodular function and let $\boldsymbol{x}$ be a vertex of $P(f)$. Furthermore, assume that $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{L}^n, \boldsymbol{a} \sqsubseteq \boldsymbol{b}$ are $\boldsymbol{x}$-tight and for all $\boldsymbol{t} \in \mathcal{L}^n$ such that $\boldsymbol{a} \sqsubset \boldsymbol{t} \sqsubset \boldsymbol{b}$ the tuple $\boldsymbol{t}$ is not $\boldsymbol{x}$-tight. Then, there is at most one coordinate $i \in [n]$ such that $\boldsymbol{a}(i) = 0_{\mathcal{L}}$ and $\boldsymbol{b}(i) = 1_{\mathcal{L}}$.*

Proof. As $\boldsymbol{x}$ is a vertex of $P(f)$ there is some $\boldsymbol{c} \in \mathbb{R}^{[n] \times \mathcal{L}}$ such that $\boldsymbol{x}$ is the unique optimum to $\max\langle \boldsymbol{c}, \boldsymbol{y} \rangle, \boldsymbol{y} \in P(f)$. Assume, for the sake of contradiction, that there are two coordinates $i, j \in [n], i \neq j$ such that $\boldsymbol{a}(i) = \boldsymbol{a}(j) = 0_{\mathcal{L}}$ and $\boldsymbol{b}(i) = \boldsymbol{b}(j) = 1_{\mathcal{L}}$. We can assume, without loss of generality, that

$$\sum_{x \in \mathcal{L}} \rho(x) \boldsymbol{c}(i, x) \geq \sum_{x \in \mathcal{L}} \rho(x) \boldsymbol{c}(j, x).$$

Let $\delta > 0$ and let the vector $\boldsymbol{x'}$ be defined by

$$\begin{array}{ll} \boldsymbol{x'}(k, x) = \boldsymbol{x}(k, x) & \text{if } k \notin \{i, j\}, \\ \boldsymbol{x'}(i, x) = \boldsymbol{x}(i, x) + \delta \cdot \rho(x) & \text{and} \\ \boldsymbol{x'}(j, x) = \boldsymbol{x}(j, x) - \delta \cdot \rho(x). & \end{array}$$

Note that $\boldsymbol{x'}$ is supermodular as $\boldsymbol{x}$ is supermodular and ρ is modular. Furthermore, $\boldsymbol{x'}(i, 0_{\mathcal{L}}) = 0$ for all $i \in [n]$ as $\boldsymbol{x}(i, 0_{\mathcal{L}}) = 0$ for all $i \in [n]$ and $\rho(0_{\mathcal{L}}) = 0$. We cannot have $\boldsymbol{x'} \in P(f)$ for any $\delta > 0$, because then either $\boldsymbol{x}$ is not the unique optimum or $\boldsymbol{x}$ is not optimal. As $\boldsymbol{x'} \notin P(f)$ there is some

$\boldsymbol{x}$-tight tuple $\boldsymbol{t} \in \mathcal{L}^n$ such that $\boldsymbol{x'}(\boldsymbol{t}) \not\leq f(\boldsymbol{t})$. It follows that $\rho(\boldsymbol{t}(j)) < \rho(\boldsymbol{t}(i))$, so $\boldsymbol{t}(i) \neq 0_\mathcal{L}$ and $\boldsymbol{t}(j) \neq 1_\mathcal{L}$. As $\boldsymbol{x}$-tight tuples are closed under meet and join (Lemma 9.10), it follows that $\boldsymbol{t'} = (\boldsymbol{b} \sqcap \boldsymbol{t}) \sqcup \boldsymbol{a}$ is $\boldsymbol{x}$-tight, which is a contradiction as $\boldsymbol{a} \sqsubset \boldsymbol{t'} \sqsubset \boldsymbol{b}$. □

By using the same technique as in the lemma above we can derive the following result.

Lemma 9.12. *Let $f : \mathcal{L}^n \to \mathbb{R}$ be a submodular function and let $\boldsymbol{x}$ be a vertex of $P(f)$. There are $\boldsymbol{x}$-tight tuples $\boldsymbol{a}$ and $\boldsymbol{b}$ such that*

- *there is at most one coordinate $i \in [n]$ which satisfies $\boldsymbol{a}(i) = 0_\mathcal{L}$; and*
- *there is at most one coordinate $i \in [n]$ which satisfies $\boldsymbol{b}(i) = 1_\mathcal{L}$.*

Proof (Sketch). We prove the first case. The second case follows analogously. By Lemma 9.10 there is a greatest $\boldsymbol{x}$-tight tuple. Let $\boldsymbol{a}$ be this tuple. Assume, for the sake of contradiction, that there are two coordinates $i, j \in [n], i \neq j$ such that $\boldsymbol{a}(i) = \boldsymbol{a}(j) = 0_\mathcal{L}$. By using the same argument as in Lemma 9.11 it follows that there is an $\boldsymbol{x}$-tight tuple $\boldsymbol{t}$ such that $\boldsymbol{a} \sqsubset \boldsymbol{a} \sqcup \boldsymbol{t}$. This contradicts the maximality of $\boldsymbol{a}$ and the lemma follows. □

Contrary to the diamond case it is not clear that $\mathbf{1}_{\mathcal{L}^n}$ is $\boldsymbol{x}$-tight for every vertex $\boldsymbol{x}$ of $P(f)$. One approach of trying to establish such a result is to, for every $i \in [n]$, try to increase $\boldsymbol{x}(i, 1_\mathcal{L})$, conclude that this is not possible as $\boldsymbol{x}$ is a vertex of $P(f)$, and get an $\boldsymbol{x}$-tight tuple $\boldsymbol{t_i}$ such that $\boldsymbol{t_i}(i) = 1_\mathcal{L}$. One can then use Lemma 9.10 to get that $\mathbf{1}_{\mathcal{L}^n}$ is $\boldsymbol{x}$-tight. However, this does not work because, in the argument above, we use the fact that there is some $\boldsymbol{c} \in \mathbb{R}^{[n]\times\mathcal{L}}$ such that $\boldsymbol{x}$ is the unique optimum to $\langle \boldsymbol{y}, \boldsymbol{c} \rangle, \boldsymbol{y} \in P(f)$. This part is not a problem. However, we also need that $\boldsymbol{c}(i, 1_\mathcal{L}) \geq 0$ for every $i \in [n]$ and this is *not* obvious. We do not need this stronger property for the results we want in this section—Lemma 9.12 is sufficient.

Lemma 9.13. *Let $\mathcal{L}$ be a modular lattice and assume that all modular lattices $\mathcal{L}'$ such that $|\mathcal{L}'| < |\mathcal{L}|$ are well-characterised. Let $f : \mathcal{L}^n \to \mathbb{Z}$ be a submodular function and let $\boldsymbol{x}$ be a vector in $\mathbb{R}^{[n]\times\mathcal{L}}$. If $\boldsymbol{x}$ is a vertex of $P(f)$, then there is a proof, which can be checked in time polynomial in n, that $\boldsymbol{x}$ is contained in $P(f)$.*

Proof. Let $h = n \cdot \rho(1_\mathcal{L})$ (h is the height of $\mathcal{L}^n$, i.e., the length of the longest chains in $\mathcal{L}$). The proof consists of a sequence $\boldsymbol{t^0}, \ldots, \boldsymbol{t^h} \in \mathcal{L}^n$ of tuples and a sequence $\boldsymbol{p^1}, \ldots, \boldsymbol{p^h}$ of "proofs" (these will be used as inputs to the verifier in Lemma 9.9). To verify the proof we check that:

1. $\boldsymbol{x}$ is supermodular and $\boldsymbol{x}(i, 0_\mathcal{L}) = 0$ for all $i \in [n]$, and
2. $\mathbf{0}_{\mathcal{L}^n} = \boldsymbol{t^0} \sqsubseteq \boldsymbol{t^1} \sqsubseteq \ldots \sqsubseteq \boldsymbol{t^{h-1}} \sqsubseteq \boldsymbol{t^h} = \mathbf{1}_{\mathcal{L}^n}$, and
3. $\boldsymbol{t^1}, \ldots, \boldsymbol{t^{h-1}}$ are $\boldsymbol{x}$-tight, and
4. for any $j \in [h]$ there it at most one coordinate $l \in [n]$ such that $\boldsymbol{t^{j-1}}(l) = 0_\mathcal{L}$ and $\boldsymbol{t^j}(l) = 1_\mathcal{L}$.

Reject the proof if any of these checks fail. We now use the verifier from Lemma 9.9 iteratively with input $(\boldsymbol{t^{i-1}}, \boldsymbol{t^i}, \boldsymbol{p^i})$, for $i \in [h]$. Reject the proof if the verifier from Lemma 9.9 rejects any input and accept it otherwise. (We cannot use Lemma 9.9 directly in the first step when we want to verify that $\boldsymbol{x}(\boldsymbol{t}) \leq f(\boldsymbol{t})$ for all $\boldsymbol{t} \in \mathcal{L}^n$ such that $\boldsymbol{t} \sqsubseteq \boldsymbol{t^1}$, as $\mathbf{0}_{\boldsymbol{\mathcal{L}^n}}$ is not necessarily $\boldsymbol{x}$-tight. However, the only place where the $\boldsymbol{x}$-tightness of $\boldsymbol{a}$ is used in Lemma 9.9 is when $\forall \boldsymbol{t} \in \mathcal{L}^n : (\boldsymbol{t} \sqsubseteq \boldsymbol{a} \vee \boldsymbol{a} \sqsubseteq \boldsymbol{t} \sqsubseteq \boldsymbol{b}) \Rightarrow \boldsymbol{x}(\boldsymbol{t}) \leq f(\boldsymbol{t})$ is shown to imply $\forall \boldsymbol{t} \in \mathcal{L}^n : \boldsymbol{t} \sqsubseteq \boldsymbol{b} \Rightarrow \boldsymbol{x}(\boldsymbol{t}) \leq f(\boldsymbol{t})$. By setting $\boldsymbol{a} = \mathbf{0}_{\boldsymbol{\mathcal{L}^n}}$ and $\boldsymbol{b} = \boldsymbol{t^1}$ we do not need this part of the proof.)

Completeness (That is, if $\boldsymbol{x}$ is a vertex of $P(f)$, then there is a proof which the verifier accepts.)

As $\boldsymbol{x}$ is a vertex of $P(f)$, there are, by Lemma 9.11 and Lemma 9.12 tuples $\boldsymbol{t^0}, \ldots, \boldsymbol{t^h}$ which make the verifier not reject the proof in steps 1–4. As $\boldsymbol{x} \in P(f)$ and all modular lattices $\mathcal{L}'$ such that $|\mathcal{L}'| < |\mathcal{L}|$ are well-characterised, there are proofs $\boldsymbol{p^1}, \ldots, \boldsymbol{p^h}$ so that the verifier will not reject the proof in the last part of the verification algorithm either.

Soundness (That is, if there is a proof which the verifier accepts, then $\boldsymbol{x} \in P(f)$.)

As the proof was not rejected in steps 1–4, the tuples $\boldsymbol{t^1}, \boldsymbol{t^2}, \ldots, \boldsymbol{t^h}$ satisfies the properties required by Lemma 9.9. Hence, by Lemma 9.9 and the assumption in the lemma, that all modular lattices $\mathcal{L}'$ such that $|\mathcal{L}'| < |\mathcal{L}|$ are well-characterised, $\boldsymbol{x}$ is contained in $P(f)$. □

Another way to state Lemma 9.13 is to say that there is an **NP** algorithm which on input $\boldsymbol{x} \in \mathbb{R}^{[n] \times \mathcal{L}}$ answers as follows:

- YES on any vertex of $P(f)$, and
- NO on any vectors not contained in $P(f)$.

Note that it is not important what the algorithm does on vectors contained in $P(f)$ which are not vertices.

The proof of the following theorem, which is the main result of this section, uses the same main ideas as Theorem 8.19 (the well-characterisedness of the diamonds). The main difference is that we do not have a min–max theorem for modular lattices, however it turns out that a min–max theorem is not actually needed to establish well-characterisedness. In the proof below we have implicitly assumed that the vertices of $P(f)$ can be encoded using polynomially many bits (measured in n). This can be shown to be the case by using the same technique as for the diamond case, i.e., using the known relations between the facet complexity and vertex complexity of polyhedrons and the definition of $P(f)$. We omit the details.

Theorem 9.14. *Every modular lattice is well-characterised.*

Proof. Let $\mathcal{L}$ be a modular lattice. We will prove the theorem by induction on $|\mathcal{L}|$. The base case, when $|\mathcal{L}| = 2$, follows as the two element lattice is oracle-tractable. Now assume that all modular lattices $\mathcal{L}'$ such that $|\mathcal{L}'| <$

$|\mathcal{L}|$ are well-characterised. We prove that $\mathcal{L}$ is well-characterised. Let $f : \mathcal{L}^n \to \mathbb{Z}$ be a submodular function and let m be some integer.

Let $N = n \cdot (|\mathcal{L}| - 1)$ (N is the dimension of $P(f)$) and let $h = n \cdot \rho(1_{\mathcal{L}})$. The proof consists of a tuple $\boldsymbol{m} \in \mathcal{L}^n$, $N+1$ vectors $\boldsymbol{x_1}, \ldots, \boldsymbol{x_{N+1}} \in \mathbb{R}^{[n]\times\mathcal{L}}$, and for each $i \in [N+1]$ a sequence $\boldsymbol{t_i^0}, \ldots, \boldsymbol{t_i^h} \in \mathcal{L}^n$ of tuples.

To verify the proof we first find $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{N+1}) \in \mathbb{R}^{N+1}$ and $\boldsymbol{y} \in \mathbb{R}^{[n]\times A}$ such that

$$\boldsymbol{y} \geq \boldsymbol{0}, \quad \sum_{i=1}^{N+1} \lambda_i \boldsymbol{x_i} = \boldsymbol{y}, \quad \boldsymbol{\lambda} \geq 0, \quad \text{and} \quad \sum_{i=1}^{N+1} \lambda_i = 1. \tag{9.9}$$

This can be done in time polynomial in n. Reject the proof if there are no solutions to (9.9). By Lemma 9.13 we can verify that $\boldsymbol{x_i} \in P(f-m)$ for each $i \in [N+1]$ (in this verification process the tuples $\boldsymbol{t_i^0}, \ldots, \boldsymbol{t_i^h}$ are used). Reject the proof if any of these vectors are not contained in $P(f-m)$. Finally accept the proof if and only if $f(\boldsymbol{m}) = m$.

Completeness (That is, if $\min_{\boldsymbol{t}\in\mathcal{L}^n} f(\boldsymbol{t}) = m$, then there is a proof which the verifier accepts.)

Let $\boldsymbol{m}$ be a tuple in $\mathcal{L}^n$ such that $\min_{\boldsymbol{t}\in\mathcal{L}^n} f(\boldsymbol{t}) = f(\boldsymbol{m}) = m$. It follows that $f(\boldsymbol{t}) - m \geq 0$ for every $\boldsymbol{t} \in \mathcal{L}^n$ and hence that $\boldsymbol{0} \in P(f-m)$. As $P(f-m)$ has vertices and the characteristic cone of $P(f-m)$ only contains vectors $\boldsymbol{c}$ such that $\boldsymbol{c} \leq \boldsymbol{0}$ it follows, from Carathéodory's theorem (see Section 8.6, in particular Theorem 8.16 for a discussion of this theorem) that there are vertices $\boldsymbol{x_1}, \ldots, \boldsymbol{x_{N+1}}$ of $P(f-m)$ such that

$$\sum_{i=1}^{N+1} \lambda_i \boldsymbol{x_i} \geq \boldsymbol{0}$$

where $\sum_{i=1}^{N+1} \lambda_i = 1$ and $\lambda_i \geq 0$ for each $i \in [N+1]$. (So some convex combination of some vertices of $P(f)$ is a vector in which each coordinate is greater than or equal to 0.) By choosing these vertices the coefficients $\lambda_1, \ldots, \lambda_{N+1}$ is a solution to (9.9). As each $\boldsymbol{x_i}$ is a vertex of $P(f)$ there is, by Lemma 9.11 and Lemma 9.12, a chain of $\boldsymbol{x_i}$-tight tuples $\boldsymbol{t_i^1} \sqsubseteq \ldots \sqsubseteq \boldsymbol{t_i^h}$ which are accepted by the verifier in Lemma 9.13.

Soundness (That is, if there is a proof which the verifier accepts, then $\min_{\boldsymbol{t}\in\mathcal{L}^n} f(\boldsymbol{t}) = m$.)

By the soundness of the verifier in Lemma 9.13 it follows that $\boldsymbol{x_i} \in P(f-m)$ for each $i \in [N+1]$. Hence, the convex combination $\boldsymbol{y}$ in (9.9) is contained in $P(f-m)$ as well. As $\boldsymbol{y} \geq 0$ it follows from Lemma 9.8 that $\boldsymbol{0} \in P(f-m)$. This means that $0 \leq f(\boldsymbol{t}) - m$ for all $\boldsymbol{t} \in \mathcal{L}^n$ and thus $\min_{\boldsymbol{t}\in\mathcal{L}^n} f(\boldsymbol{t}) \geq m$. As $f(\boldsymbol{m}) = m$ the soundness follows. □

9.4.3 Generalisations

The method in Section 9.4.2 above can be made to work on a larger class of lattices. Let $\mathcal{L}$ be a lattice and let θ be a congruence relation[2] on $\mathcal{L}$ such that $\mathcal{L}/\theta$ is modular and non-trivial (i.e., it is not the one element lattice). In this case there is a modular function $m : \mathcal{L} \to \mathbb{N}$ such that $m(0_{\mathcal{L}}) = 0$, $m(1_{\mathcal{L}}) > 0$ and m is monotone (that is, $x \sqsubseteq y$ implies $m(x) \leq m(y)$). We can construct m from the rank function ρ of $\mathcal{L}/\theta$ (which is modular as $\mathcal{L}/\theta$ is modular) as follows: $m(x) = \rho(x/\theta)$.

The function m is needed in the proof of Lemma 9.11 and in Lemma 9.12. In the induction performed in the proof of Theorem 9.14 we rely on the property that the class of lattices we are showing the result for is closed under taking intervals. That is, if we want to show the result for some class $\mathbf{C}$ of lattices, then for any $\mathcal{L} \in \mathbf{C}$ and $a, b \in \mathcal{L}, a \sqsubseteq b$ the interval $\{x \in \mathcal{L} \mid a \sqsubseteq x \sqsubseteq b\}$ must be contained in $\mathbf{C}$. These are the only two properties we use of the lattices. By imposing these two restrictions we arrive at the following result.

Theorem 9.15. *Let $\mathcal{L}$ be a lattice which has non-trivial congruence relation θ such that $\mathcal{L}/\theta$ is modular. If every proper interval of $\mathcal{L}$ is well-characterised, then $\mathcal{L}$ is well-characterised.*

Theorem 9.15 captures all modular lattices as the identity function homomorphically maps a modular lattice to a non-trivial modular lattice (itself) and any interval of a modular lattice is a modular lattice. Some non-modular lattices are also captured by the theorem, one example is the pentagon.

One might hope that Theorem 9.15 could be extended even further by constructing a non-constant modular function m for some lattice $\mathcal{L}$ which does not have any congruence such that $\mathcal{L}/\theta$ is modular and non-trivial. Unfortunately, this is not possible: Fleischer and Traynor [61] has showed that if $m : \mathcal{L} \to \mathbb{R}$ is modular, then the relation $\{(x, y) \in \mathcal{L}^2 \mid m(x) = m(y)\}$ contains the smallest congruence θ such that $\mathcal{L}/\theta$ is modular. If the only congruence θ of $\mathcal{L}$ such that $\mathcal{L}/\theta$ is modular is the trivial $\theta = \mathcal{L}^2$, then the result implies that any modular function on $\mathcal{L}$ is a constant function.

Another approach which may be useful for generalising Lemma 9.11 is the following idea: in Lemma 9.11 we are assuming, with the aim of reaching a contradiction, that we have a vertex $\boldsymbol{x}$ of $P(f)$, two $\boldsymbol{x}$-tight tuples $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{a} \sqsubset \boldsymbol{b}$, two distinct integers $i, j \in [n]$ such that $\boldsymbol{a}(i) = \boldsymbol{a}(j) = 0_{\mathcal{L}}$ and $\boldsymbol{b}(i) = \boldsymbol{b}(j) = 1_{\mathcal{L}}$. Furthermore, there is no $\boldsymbol{x}$-tight tuple $\boldsymbol{t}$ such that $\boldsymbol{a} \sqsubset \boldsymbol{t} \sqsubset \boldsymbol{b}$. We let $\boldsymbol{c} \in \mathbb{R}^{[n] \times \mathcal{L}}$ such that $\boldsymbol{x}$ is the unique solution to $\max \langle \boldsymbol{c}, \boldsymbol{y} \rangle, \boldsymbol{y} \in P(f)$. If we can show that there exist two vectors $\boldsymbol{r}^+, \boldsymbol{r}^- \in \mathbb{R}^{\mathcal{L}}$ such that

1. $\boldsymbol{r}^+$ is supermodular and $\boldsymbol{r}^-$ is submodular; and
2. $\boldsymbol{r}^+(0_{\mathcal{L}}) = \boldsymbol{r}^-(0_{\mathcal{L}}) = 0$ and $\boldsymbol{r}^+(1_{\mathcal{L}}) = \boldsymbol{r}^-(1_{\mathcal{L}}) = 1$; and

[2] See Section 10.3 for a definition of congruence relations for lattices.

3.

$$\sum_{x\in\mathcal{L}} \boldsymbol{c}(i,x)\cdot \boldsymbol{r}^{+}(x) \geq \sum_{x\in\mathcal{L}} \boldsymbol{c}(j,x)\cdot \boldsymbol{r}^{-}(x) \quad \text{or} \tag{9.10}$$

$$\sum_{x\in\mathcal{L}} \boldsymbol{c}(j,x)\cdot \boldsymbol{r}^{+}(x) \geq \sum_{x\in\mathcal{L}} \boldsymbol{c}(i,x)\cdot \boldsymbol{r}^{-}(x), \tag{9.11}$$

then we can define the vector $\boldsymbol{x'}$ by

$$\begin{array}{ll} \boldsymbol{x'}(k,x) = \boldsymbol{x}(k,x) & \text{if } k \notin \{i,j\}, \\ \boldsymbol{x'}(i,x) = \boldsymbol{x}(i,x) + \delta\cdot \boldsymbol{r}^{+}(x) & \text{and,} \\ \boldsymbol{x'}(j,x) = \boldsymbol{x}(j,x) - \delta\cdot \boldsymbol{r}^{-}(x) & \end{array}$$

if (9.10) holds and

$$\begin{array}{ll} \boldsymbol{x'}(k,x) = \boldsymbol{x}(k,x) & \text{if } k \notin \{i,j\}, \\ \boldsymbol{x'}(i,x) = \boldsymbol{x}(i,x) - \delta\cdot \boldsymbol{r}^{-}(x) & \text{and,} \\ \boldsymbol{x'}(j,x) = \boldsymbol{x}(j,x) + \delta\cdot \boldsymbol{r}^{+}(x) & \end{array}$$

otherwise (if (9.11) holds). For a sufficiently small $\delta > 0$ the vector $\boldsymbol{x'}$ is contained in $P(f)$. Furthermore, by (9.10) (or (9.11)) it follows that $\langle \boldsymbol{x}, \boldsymbol{c}\rangle \leq \langle \boldsymbol{x'}, \boldsymbol{c}\rangle$, which contradicts our choice of $\boldsymbol{c}$. From this we conclude that for every vertex $\boldsymbol{x}$ of $P(f)$ and $\boldsymbol{x}$-tight tuples $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{a} \sqsubset \boldsymbol{b}$ either there is some $\boldsymbol{x}$-tight tuple $\boldsymbol{t}$ such that $\boldsymbol{a} \sqsubset \boldsymbol{t} \sqsubset \boldsymbol{b}$ or there is at most one coordinate $i \in [n]$ such that $\boldsymbol{a}(i) = 0_{\mathcal{L}}$ and $\boldsymbol{b}(i) = 1_{\mathcal{L}}$. So the problem of establishing Lemma 9.11 for a lattice $\mathcal{L}$ can be solved by showing that for every $\boldsymbol{c} \in \mathbb{R}^{[n]\times\mathcal{L}}$ there are vectors $\boldsymbol{r}^{+}$ and $\boldsymbol{r}^{-}$ which satisfy the conditions 1–3.

Note that if $\mathcal{L}$ is modular, then we can let $\boldsymbol{r}^{+}(x) = \boldsymbol{r}^{-}(x) = \rho(x)/\rho(1_{\mathcal{L}})$, where ρ is the rank function of $\mathcal{L}$. As $\rho(0_{\mathcal{L}}) = 0$ and ρ is modular condition 1 and 2 are satisfied. Furthermore, as $\boldsymbol{r}^{+} = \boldsymbol{r}^{-}$ at least one of (9.10) and (9.11) holds. This is essentially what we did in the proof of Lemma 9.11.

9.5 Obstacles to Oracle-pseudo-tractability

There is essentially only one reason for why the oracle-pseudo-tractability proofs of Chapter 8 cannot directly be extended to work for general modular atomistic lattices. The problem is Theorem 8.26 which states that the vertices of $P_M(f)$ are half-integral. A crucial part of the proof of Theorem 8.26 is Lemma 8.24 which gives a structure result for the vertices of $P_M(f)$. We do not have such a structure result for the vertices of $P(f)$ for general modular atomistic lattices. A starting point may be to show that the vertices of $P(f)$ satisfies some of the properties of unified vectors. In particular, do the vertices of $P(f)$ satisfy conditions 1, 3, and 4 in Definition 9.1? If yes, then this would be somewhat similar to Lemma 8.24.

If the same algorithmic framework is to be used we would need something similar to the half-integrality result for the modular atomistic case. In

particular, something along the lines of "For each modular atomistic lattice $\mathcal{L}$ there is a $k \in \mathbb{N}$ such that for any submodular $f : \mathcal{L}^n \to \mathbb{Z}$ the polyhedron $P(f)$ is $1/k$-integral" would be needed. Here "$1/k$-integral" means that if $\boldsymbol{x}$ is a vertex of $P(f)$, then every coordinate of $\boldsymbol{x}$ is contained in $\{1/k \cdot m \mid m \in \mathbb{Z}\}$. We have gained some weak support for this hypothesis by some very limited computational experiments, but it is not enough to state it as a conjecture.

The other parts of the argument in Section 8.7 should work without much changes. In particular, it is sufficient to consider strictly submodular functions as $\boldsymbol{t} \to \rho(\boldsymbol{t})(\rho(\mathbf{1}_{\mathcal{L}^n}) - \rho(\boldsymbol{t}))$ is strictly submodular (the proof of Lemma 8.21 works essentially unmodified) and the maximum value of this function grows polynomially with n.

For modular lattices the issues above apply as well. However, slightly more work remains as we have not shown that the greedy algorithm can be used to find an initial vertex of $P(f)$ which we then can improve upon.

Chapter 10

Structure in Tractable Classes of SFM

10.1 Introduction

In this chapter we will prove that the known classes of lattices which are oracle-tractable (oracle-pseudo-tractable, well-characterised) are closed under various operations. In particular, we will show that these classes are closed under taking sublattices, homomorphic images, finite direct products and Mal'tsev products (homomorphic images and Mal'tsev products will be defined in Section 10.3 below). Classes of lattices (or algebras in general) that are closed under the three first operations are called *pseudovarieties*. Pseudovarieties should not be confused with *varieties* which are classes of lattices (or algebras) which are closed under sublattices, homomorphic images and general, not necessarily finite, direct products.

10.2 Sublattices of Modular Lattices

In this section we will show that for any modular lattice $\mathcal{L}$ and sublattice $\mathcal{X}$ of $\mathcal{L}$, if $\mathcal{L}$ is oracle-tractable (oracle-pseudo-tractable, well-characterised), then so is $\mathcal{X}$. This is proved by modifying a construction due to Schrijver [133]. We will also give an algorithm for the more general problem when $\mathcal{S}$ is a sublattice of $\mathcal{L}^n$ and we are given a submodular function $f : \mathcal{S} \to \mathbb{R}$ which we want to minimise in time polynomial in n.

As mentioned in Chapter 7 Schrijver [133] proved that given a finite set V, for any sublattice S of 2^V and submodular function $f : S \to \mathbb{R}$ a minimum of f can be found in time polynomial in $|V|$. (Certain mild additional assumptions are also needed, in particular for each $v \in V$ we need to know the smallest $X_v \in S$ such that $v \in X_v$. We also need to know the smallest set X in S.) In this section we adapt Schrijver's approach to

work for general modular lattices.

Given a modular lattice $\mathcal{L}$, a sublattice $\mathcal{S}$ of $\mathcal{L}^n$, and a submodular function $f : \mathcal{S} \to \mathbb{R}$ we want to minimise f in time polynomial in n. To do this we must have some information about $\mathcal{S}$. In the algorithm below we need to compute a certain operation, which depends on $\mathcal{S}$, on the elements of $\mathcal{L}^n$ efficiently. In particular, given $\boldsymbol{x} \in \mathcal{L}^n$ we must be able to find the minimal $\boldsymbol{x'} \in \mathcal{S}$ such that $\boldsymbol{x} \sqsubseteq \boldsymbol{x'}$ in time polynomial in n. In the approach taken below we also need to be supplied with an upper bound on $\max_{\boldsymbol{y}\in\mathcal{S}} |f(\boldsymbol{y})|$. In the case when $\mathcal{S} = \mathcal{X}^n$ for some sublattice $\mathcal{X}$ of $\mathcal{L}$, we do not need this upper bound, we state and prove this latter result as Theorem 10.2.

Theorem 10.1. *Let n be a positive integer, $\mathcal{L}$ a modular lattice, $\mathcal{S}$ a sublattice of $\mathcal{L}^n$, and $f : \mathcal{S} \to \mathbb{Z}$ a submodular function such that $\max_{\boldsymbol{y}\in\mathcal{S}} |f(\boldsymbol{y})| \leq c$ for some $c \in \mathbb{N}$.*

1. *If SFM($\mathcal{L}$) is oracle-tractable, then there is an algorithm which finds a minimum of f in time polynomial in n and $\log c$.*
2. *If SFM($\mathcal{L}$) is oracle-pseudo-tractable, then there is an algorithm which finds a minimum of f in time polynomial in n and c.*
3. *If SFM($\mathcal{L}$) is well-characterised and $\min_{\boldsymbol{y}\in\mathcal{S}} f(\boldsymbol{y}) = m$, then there is a proof of this fact which can be checked in time polynomial in n and $\log c$.*

In each case it is assumed that c is a part of the input to the algorithm.

Proof. Let $f : \mathcal{S} \to \mathbb{Z}$ be a submodular function and let c be the upper bound on $\max_{\boldsymbol{y}\in\mathcal{S}} |f(\boldsymbol{y})|$. Let $d = 2c$ and let $\rho : \mathcal{L} \to \mathbb{N}$ be the rank function for $\mathcal{L}$. By abuse of notation we will use ρ as the rank function for $\mathcal{L}^n$ as well. As ρ is a rank function we have $\rho(\boldsymbol{x}) < \rho(\boldsymbol{y})$ whenever $\boldsymbol{x} \sqsubset \boldsymbol{y}$. For all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$ with $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$ we have

$$f(\boldsymbol{y}) + d \cdot \rho(\boldsymbol{y}) \geq f(\boldsymbol{x}) + d \cdot \rho(\boldsymbol{x}). \tag{10.1}$$

This follows from the properties of the rank function and our choice of d. For any $\boldsymbol{x} \in \mathcal{L}^n$ we let $\overline{\boldsymbol{x}}$ denote the minimal element in $\mathcal{S}$ which satisfies $\boldsymbol{x} \sqsubseteq \overline{\boldsymbol{x}}$. Define $g : \mathcal{L}^n \to \mathbb{R}$ by

$$g(\boldsymbol{x}) = f(\overline{\boldsymbol{x}}) + d \cdot \rho(\overline{\boldsymbol{x}}).$$

We claim that g is submodular. Let $\boldsymbol{x}, \boldsymbol{y}$ be two arbitrary tuples in $\mathcal{L}^n$, then

$$\begin{aligned}
g(\boldsymbol{x}) + g(\boldsymbol{y}) &= f(\overline{\boldsymbol{x}}) + d \cdot \rho(\overline{\boldsymbol{x}}) + f(\overline{\boldsymbol{y}}) + d \cdot \rho(\overline{\boldsymbol{y}}) \\
&\geq f(\overline{\boldsymbol{x}} \sqcap \overline{\boldsymbol{y}}) + f(\overline{\boldsymbol{x}} \sqcup \overline{\boldsymbol{y}}) + d \cdot \rho(\overline{\boldsymbol{x}}) + d \cdot \rho(\overline{\boldsymbol{y}}) \\
&= f(\overline{\boldsymbol{x}} \sqcap \overline{\boldsymbol{y}}) + f(\overline{\boldsymbol{x}} \sqcup \overline{\boldsymbol{y}}) + d \cdot \rho(\overline{\boldsymbol{x}} \sqcap \overline{\boldsymbol{y}}) + d \cdot \rho(\overline{\boldsymbol{x}} \sqcup \overline{\boldsymbol{y}}) \\
&\geq f(\overline{\boldsymbol{x} \sqcap \boldsymbol{y}}) + f(\overline{\boldsymbol{x} \sqcup \boldsymbol{y}}) + d \cdot \rho(\overline{\boldsymbol{x} \sqcap \boldsymbol{y}}) + d \cdot \rho(\overline{\boldsymbol{x} \sqcup \boldsymbol{y}}) \\
&= g(\boldsymbol{x} \sqcap \boldsymbol{y}) + g(\boldsymbol{x} \sqcup \boldsymbol{y})
\end{aligned}$$

where the first inequality follows from the submodularity of f and the second inequality from (10.1) and Claim A and B below.

Claim A. $\overline{\boldsymbol{x}} \sqcup \overline{\boldsymbol{y}} \sqsupseteq \overline{\boldsymbol{x} \sqcup \boldsymbol{y}}$.

Proof. We have $\overline{\boldsymbol{x}} \sqcup \overline{\boldsymbol{y}} \sqsupseteq \boldsymbol{x} \sqcup \boldsymbol{y}$, furthermore as $\mathcal{S}$ is a lattice and $\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}} \in \mathcal{S}$ we have $\overline{\boldsymbol{x}} \sqcup \overline{\boldsymbol{y}} \in \mathcal{S}$. It follows that the least element in $\mathcal{S}$, greater than or equal to $\boldsymbol{x} \sqcup \boldsymbol{y}$, must be less than or equal to $\overline{\boldsymbol{x}} \sqcup \overline{\boldsymbol{y}}$. In other words, $\overline{\boldsymbol{x}} \sqcup \overline{\boldsymbol{y}} \sqsupseteq \overline{\boldsymbol{x} \sqcup \boldsymbol{y}}$.

Claim B. $\overline{\boldsymbol{x}} \sqcap \overline{\boldsymbol{y}} \sqsupseteq \overline{\boldsymbol{x} \sqcap \boldsymbol{y}}$.

Proof. It is clear that $\overline{\boldsymbol{x}} \sqsupseteq \overline{\boldsymbol{x} \sqcap \boldsymbol{y}}$ and $\overline{\boldsymbol{y}} \sqsupseteq \overline{\boldsymbol{x} \sqcap \boldsymbol{y}}$. The claim follows.

As g is submodular on $\mathcal{L}^n$ the function $h(\boldsymbol{t}) = g(\boldsymbol{t}) - d \cdot \rho(\boldsymbol{t})$ is submodular as well (as ρ is modular and g is submodular this function is submodular). Let $\boldsymbol{t} \in \mathcal{L}^n$ be a minimiser of h. For any $\boldsymbol{x} \in \mathcal{S}$ we have

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) - d \cdot \rho(\boldsymbol{x}) \geq g(\boldsymbol{t}) - d \cdot \rho(\boldsymbol{t}) = f(\overline{\boldsymbol{t}}) + d \cdot \rho(\overline{\boldsymbol{t}}) - d \cdot \rho(\boldsymbol{t}) \geq f(\overline{\boldsymbol{t}}).$$

The first inequality follows from the fact that $\boldsymbol{t}$ is a minimiser of $g(\boldsymbol{t}) - d \cdot \rho(\boldsymbol{t})$ and the second one from the monotonicity of ρ. We conclude that $\overline{\boldsymbol{t}}$ minimises f over $\mathcal{S}$.

We are now ready to prove the three cases of the theorem. As for Case 1, if SFM($\mathcal{L}$) is oracle-tractable, then we can find a tuple $\boldsymbol{t}$ which minimises h in time polynomial in n and $\log c$. By the argument above $\overline{\boldsymbol{t}}$ is a minimiser of f. Case 2 is very similar, the only difference being that the running time now is polynomial in n and c instead.

We now turn to Case 3. Choose $\boldsymbol{t} \in \mathcal{L}^n$ and $m \in \mathbb{Z}$ so that

$$\min_{\boldsymbol{y} \in \mathcal{L}^n} h(\boldsymbol{y}) = h(\boldsymbol{t}) = m. \tag{10.2}$$

As SFM($\mathcal{L}$) is well-characterised, given $\boldsymbol{t}$ and m, there are proofs which can be checked in time polynomial in n and $\log c$ that (10.2) is true. By the argument above $\overline{\boldsymbol{t}}$ is a minimiser of f and hence the minimum value attained by f is $f(\overline{\boldsymbol{t}})$. □

When the sublattice is of the form $\mathcal{X}^n$ where $\mathcal{X}$ is some sublattice of $\mathcal{L}$, we can compute an appropriate c from f. This is captured by the following theorem, which also based on [133].

Theorem 10.2. *Let $\mathcal{L}$ be a modular lattice such that SFM($\mathcal{L}$) is oracle-tractable (oracle-pseudo-tractable, well-characterised). If $\mathcal{X}$ is a sublattice of $\mathcal{L}$, then SFM($\mathcal{X}$) is oracle-tractable (oracle-pseudo-tractable, well-characterised).*

To prove the theorem we show how one can compute a $d \in \mathbb{Z}$ which is sufficiently large for the algorithm in Theorem 10.1 to work and at the same time not too large to break our time budget. The technique used in the proof below is the same as we used in Section 8.5 to compute an upper bound of $\max(|f|)$ in polynomial time.

Proof. Let $f : \mathcal{X}^n \to \mathbb{R}$ be submodular. Let

$$d' = \max\{f(\mathbf{1}_{\boldsymbol{\mathcal{X}}^n}[i = x]) - f(\mathbf{1}_{\boldsymbol{\mathcal{X}}^n}[i = y]) \mid i \in [n], x, y \in \mathcal{X}, x \prec y\}$$

and let $d = \max\{d', 0\}$. Note that we can compute d in time polynomial in n. Let ρ be the rank function for $\mathcal{L}^n$. For all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}^n$ with $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$ we claim that

$$f(\boldsymbol{x}) + d \cdot \rho(\boldsymbol{x}) \leq f(\boldsymbol{y}) + d \cdot \rho(\boldsymbol{y}). \tag{10.3}$$

We prove that $\boldsymbol{x} \prec \boldsymbol{y}$ implies $f(\boldsymbol{x}) \leq f(\boldsymbol{y}) + d$, this is sufficient to establish (10.3). As $\boldsymbol{x} \prec \boldsymbol{y}$ we have $\boldsymbol{x} = \boldsymbol{y} \sqcap \mathbf{1}_{\boldsymbol{\mathcal{L}}^n}[i = \boldsymbol{x}(i)]$ for some $i \in [n]$. We now get

$$\begin{aligned} f(\boldsymbol{x}) &= f(\boldsymbol{y} \sqcap \mathbf{1}_{\boldsymbol{\mathcal{X}}^n}[i = \boldsymbol{x}(i)]) \\ &\leq f(\boldsymbol{y}) + f(\mathbf{1}_{\boldsymbol{\mathcal{X}}^n}[i = \boldsymbol{x}(i)]) - f(\boldsymbol{y} \sqcup \mathbf{1}_{\boldsymbol{\mathcal{X}}^n}[i = \boldsymbol{x}(i)]) \\ &= f(\boldsymbol{y}) + f(\mathbf{1}_{\boldsymbol{\mathcal{X}}^n}[i = \boldsymbol{x}(i)]) - f(\mathbf{1}_{\boldsymbol{\mathcal{X}}^n}[i = \boldsymbol{y}(i)]) \\ &\leq f(\boldsymbol{y}) + d. \end{aligned}$$

Here the first inequality follows from the submodularity of f and the second equality follows from the fact that $\boldsymbol{x}(i) \sqsubseteq \boldsymbol{y}(i)$. The last inequality follows from our choice of d. The constant d, which satisfies (10.3), is large enough to make the algorithm in Theorem 10.1 work (in particular (10.1) is satisfied by our d). Furthermore, d is not too large compared to $\max(|f|)$, indeed $d \leq 2\max(|f|)$. We conclude that the running time of the algorithm in Theorem 10.1 is polynomial in n. □

10.3 Other Constructions on Lattices

In addition to the sublattice construction of modular lattices, which we proved preserved non-hardness results in the previous section, we will see that two other lattice constructions also preserve non-hardness. We first define the two constructions we will work with, homomorphic images and Mal'tsev products.

Definition 10.3 (Homomorphism of lattices)**.** *Given two lattices* $\mathcal{A} = (A; \sqcap_A, \sqcup_A)$ *and* $\mathcal{B} = (B; \sqcap_B, \sqcup_B)$*, then* $\phi : A \to B$ *is said to be a* homomorphism *from* $\mathcal{A}$ *to* $\mathcal{B}$ *if*

$$\phi(x \sqcap_A y) = \phi(x) \sqcap_B \phi(x) \quad \textit{and} \quad \phi(x \sqcup_A y) = \phi(x) \sqcup_B \phi(x)$$

for all $x, y \in A$*. If* ϕ *is surjective, then* $\mathcal{B}$ *is a* homomorphic image *of* $\mathcal{A}$*.*

Given a homomorphism ϕ mapping $\mathcal{A} = (A; \sqcap_A, \sqcup_A)$ to $\mathcal{B} = (B; \sqcap_B, \sqcup_B)$, we can construct an equivalence relation θ on A as $\theta = \{(x, y) \mid \phi(x) = \phi(y)\}$. The relation θ is said to be a *congruence relation* of $\mathcal{A}$. We can now construct the *quotient lattice* $\mathcal{A}/\theta = (A/\theta; \sqcap_A/\theta, \sqcup_A/\theta)$. Here, $A/\theta = \{x/\theta \mid$

$x \in A\}$ and x/θ is the equivalence class containing x. Furthermore, $\sqcap_A/\theta$ and $\sqcup_A/\theta$ are defined as $x/\theta \ \sqcap_A /\theta \ y/\theta = (x \sqcap_A y)/\theta$ and $x/\theta \ \sqcup_A /\theta \ y/\theta = (x \sqcup_A y)/\theta$, respectively. It is well-known that if ϕ is surjective, then $\mathcal{A}/\theta$ is isomorphic to $\mathcal{B}$. [27]

The following result was proved for oracle-tractability by Krokhin and Larose [109]. Essentially the same proof can be used to derive the following result.

Theorem 10.4. *Let $\mathcal{L} = (L; \sqcap_L, \sqcup_L)$ be a finite lattice which is oracle-tractable (oracle-pseudo-tractable, well-characterised). If $\mathcal{H} = (H; \sqcap_H, \sqcup_H)$ is a homomorphic image of $\mathcal{L}$, then also $\mathcal{H}$ is oracle-tractable (oracle-pseudo-tractable, well-characterised).*

Proof. Let $\phi : \mathcal{L} \to \mathcal{H}$ be the surjective homomorphism from $\mathcal{L}$ to $\mathcal{H}$ and let $f : \mathcal{H}^n \to \mathbb{Z}$ be submodular. Define $g : \mathcal{L}^n \to \mathbb{Z}$ as $g(\boldsymbol{x}) = f(\phi(\boldsymbol{x}))$ (we are evaluating ϕ componentwise here). It is not hard to show that g is submodular. Indeed, for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{L}^n$ we have

$$\begin{aligned} g(\boldsymbol{x}) + g(\boldsymbol{y}) &= f(\phi(\boldsymbol{x})) + f(\phi(\boldsymbol{y})) \\ &\geq f(\phi(\boldsymbol{x}) \sqcup_H \phi(\boldsymbol{y})) + f(\phi(\boldsymbol{x}) \sqcap_H \phi(\boldsymbol{y})) \\ &= f(\phi(\boldsymbol{x} \sqcup_L \boldsymbol{y})) + f(\phi(\boldsymbol{x} \sqcap_L \boldsymbol{y})) \\ &= g(\boldsymbol{x} \sqcup_L \boldsymbol{y}) + g(\boldsymbol{x} \sqcap_L \boldsymbol{y}). \end{aligned}$$

Here the inequality is the submodularity of f and the second equality follows as ϕ is a homomorphism. As ϕ is surjective $\min_{\boldsymbol{x} \in \mathcal{H}^n} f(\boldsymbol{x})$ coincides with $\min_{\boldsymbol{x} \in \mathcal{L}^n} g(\boldsymbol{x})$. Furthermore, if $\boldsymbol{y}$ is a minimiser of g, then $\phi(\boldsymbol{y})$ is a minimiser of f.

If $\mathcal{L}$ is oracle-tractable (or oracle-pseudo-tractable), then clearly $\mathcal{H}$ is oracle-tractable (or oracle-pseudo-tractable) as well as we can minimise g to find the minimum of f. If $\mathcal{L}$ is well-characterised, then $\mathcal{H}$ is well-characterised as well. We simply use the well-characterisedness of $\mathcal{L}$ to obtain a proof of $\min_{\boldsymbol{x} \in \mathcal{L}^n} g(\boldsymbol{x}) = m$ and as the minimum of f and g coincide this is also a proof that $\min_{\boldsymbol{x} \in \mathcal{H}^n} f(\boldsymbol{x}) = m$. □

Krokhin and Larose [109] also investigated the *Mal'tsev product* of classes of lattices and proved that it preserves oracle-tractability. Mal'tsev products have been studied extensively in lattice theory [70, 71] and for other types of algebras [114].

Definition 10.5 (Mal'tsev product)**.** *Let $\mathbf{X}$ and $\mathbf{Y}$ be two classes of lattices. A lattice $\mathcal{L}$ is contained in the* Mal'tsev product $\mathbf{X} \circ \mathbf{Y}$ *if there is a congruence θ on $\mathcal{L}$ such that $\mathcal{L}/\theta \in \mathbf{Y}$ and every θ-class is contained in $\mathbf{X}$.*

Note that if $\mathbf{A}$ and $\mathbf{B}$ are classes of lattices and $\mathcal{L}_1 \in \mathbf{A}, \mathcal{L}_2 \in \mathbf{B}$, then $\mathcal{L}_1 \times \mathcal{L}_2 \in \mathbf{A} \circ \mathbf{B}$. (We can define a congruence θ on $\mathcal{L}_1 \times \mathcal{L}_2$ by saying that $(x, y), (x', y') \in \mathcal{L}_1 \times \mathcal{L}_2$ are θ-related if and only if $y = y'$. This was also observed in [109].) Hence, Mal'tsev products are more general than direct products. Figure 10.1 contains two examples of Mal'tsev products. In [109] the following result was shown.

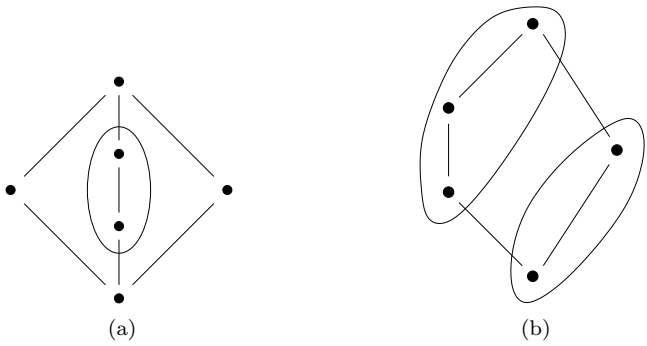


Figure 10.1: Examples of Mal'tsev products. The lattice in (a) is contained in $\{\mathcal{C}_2\} \circ \{\mathcal{M}_3\}$ and the lattice in (b) is the pentagon and is contained in $\{\mathcal{C}_2, \mathcal{C}_3\} \circ \{\mathcal{C}_2\}$. The congruence θ is chosen so that elements within an ellipse are θ-related and elements not contained in any ellipse are only θ-related to themselves. Here $\mathcal{C}_2$ is the two element chain, $\mathcal{C}_3$ is the three element chain and $\mathcal{M}_3$ is the diamond with three atoms. Before the results in Chapter 8 were obtained no non-hardness results were known for the lattice in (a).

Theorem 10.6. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *are oracle-tractable classes of lattices, then* $\mathbf{A} \circ \mathbf{B}$ *is oracle-tractable.*

By using essentially the same proof as for the theorem above one obtains the following result.

Theorem 10.7. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *are oracle-tractable (oracle-pseudo-tractable, well-characterised) classes of lattices, then* $\mathbf{A} \circ \mathbf{B}$ *is oracle-tractable (oracle-pseudo-tractable, well-characterised).*

Proof. Let n be a positive integer and let $f : \mathcal{L} \to \mathbb{Z}$ be submodular, where $\mathcal{L} = \mathcal{L}_1 \times \mathcal{L}_2 \times \ldots \times \mathcal{L}_n$ and $\mathcal{L}_i \in \mathbf{A} \circ \mathbf{B}$ for $i \in [n]$. As $\mathcal{L}_i \in \mathbf{A} \circ \mathbf{B}$ there are congruence relations $\theta_1, \ldots, \theta_n$ such that $\mathcal{L}_1/\theta_1, \ldots, \mathcal{L}_n/\theta_n \in \mathbf{B}$. Furthermore, for each $i \in [n]$ each θ_i-class is contained in $\mathbf{A}$. For $i \in [n]$ let $\mathcal{H}_i$ denote $\mathcal{L}/\theta_i$ and let $\mathcal{H} = \mathcal{H}_1 \times \ldots \times \mathcal{H}_n$. For a tuple $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{L}$ we will use the notation $\boldsymbol{x}/\theta$ to denote the tuple $(x_1/\theta_1, x_2/\theta_2, \ldots, x_n/\theta_n) \in \mathcal{H}$. Let $g : \mathcal{H} \to \mathbb{Z}$ be defined as

$$g(a_1/\theta_1, a_2/\theta_2, \ldots, a_n/\theta_n) = \\ \min \{f(b_1, b_2, \ldots, b_n) \mid \forall i \in [n] : b_i \in a_i/\theta_i\}.$$

We claim that g is submodular. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{L}$ such that $g(\boldsymbol{x}/\theta) = f(\boldsymbol{x})$ and

$g(\boldsymbol{y}/\theta) = f(\boldsymbol{y})$. We now get

$$\begin{aligned} g(\boldsymbol{x}/\theta) + g(\boldsymbol{y}/\theta) &= f(\boldsymbol{x}) + f(\boldsymbol{y}) \\ &\geq f(\boldsymbol{x} \sqcup \boldsymbol{y}) + f(\boldsymbol{x} \sqcap \boldsymbol{y}) \\ &\geq g((\boldsymbol{x} \sqcup \boldsymbol{y})/\theta) + g((\boldsymbol{x} \sqcap \boldsymbol{y})/\theta) \\ &= g(\boldsymbol{x}/\theta \sqcup \boldsymbol{y}/\theta) + g(\boldsymbol{x}/\theta \sqcap \boldsymbol{y}/\theta). \end{aligned}$$

Here the first equality follows from our choice of $\boldsymbol{x}$ and $\boldsymbol{y}$. The first inequality is the submodularity of f. The second inequality follows from the definition of g and the fact that $\boldsymbol{x} \sqcup \boldsymbol{y} \in (\boldsymbol{x} \sqcup \boldsymbol{y})/\theta$ and $\boldsymbol{x} \sqcap \boldsymbol{y} \in (\boldsymbol{x} \sqcap \boldsymbol{y})/\theta$. The final equality follows as θ is a congruence relation. Note that it is clear that $\min_{\boldsymbol{x} \in \mathcal{L}} f(\boldsymbol{x})$ coincides with $\min_{\boldsymbol{x} \in \mathcal{H}} g(\boldsymbol{x})$. Hence, it is sufficient to minimise g.

We will first prove the theorem for oracle-tractability and oracle-pseudo-tractability. As for each $i \in [n]$ each θ_i-class is oracle-tractable (oracle-pseudo-tractable) it follows that for a given tuple $(x_1/\theta_1, \ldots, x_n/\theta_n)$ we can compute $g(x_1/\theta_1, \ldots, x_n/\theta_n)$ in polynomial time (pseudopolynomial time) by minimising f (which is submodular) over the sublattice $x_1/\theta_1 \times \ldots \times x_n/\theta_n$ as $x_i/\theta_i \in \mathbf{A}$ for $i \in [n]$. As $\mathcal{H}_i$ is oracle-tractable (oracle-pseudo-tractable) it follows that there is some algorithm for minimising g which runs in polynomial time (pseudopolynomial time). From time to time the algorithm will use the oracle to evaluate g on some tuple. As each such oracle request can be computed in polynomial time (pseudopolynomial time), as argued above, it follows that the minimum of g can be computed in polynomial time (pseudopolynomial time). As the minimum of g coincides with the minimum of f the algorithm is correct.

We will now show the implication in the theorem when $\mathbf{A}$ and $\mathbf{B}$ are well-characterised. The proof which the verifier will use consists of two parts:

- a pair $(m, \boldsymbol{p})$ where m is an integer and $\boldsymbol{p}$ is some abstract object, a proof; and
- a positive integer k and a set of 3-tuples

  $$\{(\boldsymbol{x_1}/\theta, m_1, \boldsymbol{p_1}), (\boldsymbol{x_2}/\theta, m_2, \boldsymbol{p_2}), \ldots, (\boldsymbol{x_k}/\theta, m_k, \boldsymbol{p_k})\}$$

  where each 3-tuple $(\boldsymbol{x_i}/\theta, m_i, \boldsymbol{p_i})$ consists of a tuple $\boldsymbol{x_i}/\theta \in \mathcal{H}$, an integer m_i, and a proof $\boldsymbol{p_i}$.

As $\{x/\theta_i \mid x \in \mathcal{L}_i\} \subseteq \mathbf{A}$ for $i \in [n]$ is well-characterised there is a polynomial time verification algorithm V for the union of these sets of lattices. Furthermore, as $\{\mathcal{H}_1, \ldots, \mathcal{H}_n\}$ is well-characterised there is a verification algorithm V_H for this set of lattices. To verify the proof the verifier proceeds as follows:

1. Verify that $g(\boldsymbol{x_i}/\theta) = m_i$ for all $(\boldsymbol{x_i}/\theta, m_i, \boldsymbol{p_i}) \in P$ using the proof $\boldsymbol{p_i}$ and V. Reject the proof if any of these checks fail.

2. Use $\boldsymbol{p}$ and V_H to verify that $\min_{\boldsymbol{x}\in\mathcal{H}} g(\boldsymbol{x}) = m$. During the verification process V_H will evaluate g from time to time. If V_H tries to evaluate g on some tuple $\boldsymbol{x}$ and $\boldsymbol{x} = \boldsymbol{x_i}$ for some $i \in [k]$, then m_i is returned to V_H. If there is no such i, the proof is rejected.

3. The proof is accepted if and only if $\boldsymbol{p}$ is accepted by V_H.

Completeness. Let $m = \min_{\boldsymbol{x}\in\mathcal{H}} g(\boldsymbol{x})$. As $\mathcal{H}_1, \ldots, \mathcal{H}_n$ are well-characterised, there is a proof $\boldsymbol{p}$ such that V_H will accept on input $(m, \boldsymbol{p})$. For a tuple $\boldsymbol{x}/\theta = (x_1/\theta_1, \ldots, x_n/\theta_n) \in \mathcal{H}$ let $\boldsymbol{p}(\boldsymbol{x}/\theta)$ be a proof which is accepted by V to verify the claim that

$$\min \{f(y_1, y_2, \ldots, y_n) \mid \forall i \in [n] : y_i \in x_i/\theta_i\} \tag{10.4}$$

equals the integer $g(\boldsymbol{x}/\theta)$. As $x_i/\theta_i \in \mathbf{A}$ for $i \in [n]$ and (10.4) equals $g(\boldsymbol{x}/\theta)$ there is such a proof $\boldsymbol{p}(\boldsymbol{x}/\theta)$. Let $T \subseteq \mathcal{H}$ be the set of all tuples on which V_H, with input $\boldsymbol{p}$, evaluates g. Choose P to be the set

$$\{(\boldsymbol{x}/\theta, g(\boldsymbol{x}/\theta), \boldsymbol{p}(\boldsymbol{x}/\theta)) \mid \boldsymbol{x}/\theta \in T\}.$$

By our choice of P the verifier will not reject the proof in Step 1. Furthermore, as T contains all tuples on which V_H evaluates g the verifier will not reject the proof in Step 2. Finally, as $\min_{\boldsymbol{x}\in\mathcal{H}} g(\boldsymbol{x}) = m$, the verifier will not reject the proof in Step 3 either and hence it is accepted.

Soundness. As the proof was not rejected in Step 1 and as V is sound it follows that $g(\boldsymbol{x_i}/\theta) = m_i$ for all $(\boldsymbol{x_i}/\theta, m_i, \boldsymbol{p_i}) \in P$. Furthermore, as the proof was not rejected in Step 2 nor in Step 3 and as V_H is sound it follows that $\min_{\boldsymbol{x}\in\mathcal{H}} g(\boldsymbol{x}) = m$.

This concludes the proof of the theorem. □

It is not known if one can impose restrictions on the submodular functions in Theorem 10.7. In particular, as the theorem is stated above, we cannot assume that the submodular functions can be expressed as a sum of predicates. Recall that $\text{Spmod}_{\mathcal{L}}$ denotes the class of all supermodular relations over the lattice $\mathcal{L}$. In Section 7.5 we saw that the objective function of Max CSP($\text{Spmod}_{\mathcal{L}}$) is a sum of predicates which are supermodular on $\mathcal{L}$. As shown in Theorem 7.7 oracle-tractability (oracle-pseudo-tractability, well-characterisedness) implies containment in **PO** (pseudopolynomial-time algorithm, containment in **coNP**) for W-Max CSP($\text{Spmod}_{\mathcal{L}}$). One could hope that Theorem 10.7 could be modified to something along the lines of:

> Let $\mathbf{A}$ and $\mathbf{B}$ be classes of lattices such that W-Max CSP($\text{Spmod}_{\mathcal{L}}$) is in **PO** for any $\mathcal{L} \in \mathbf{A} \cup \mathbf{B}$. Then, W-Max CSP($\text{Spmod}_{\mathcal{X}}$) is in **PO** for any lattice $\mathcal{X} \in \mathbf{A} \circ \mathbf{B}$.

However, such an implication is not known to hold. What this means is that tractability results for Max CSP($\text{Spmod}_{\mathcal{L}}$) for some lattice $\mathcal{L}$ does not immediately imply tractability for other lattices via a Mal'tsev product construction. Recall that it is known that W-Max CSP($\text{Spmod}_{\mathcal{M}_k}$) is in **PO**

for all $k \geq 3$. [109] Although $\text{Spmod}_{\mathcal{M}_k}$ and $\text{Spmod}_{\mathcal{C}_2}$ (here $\mathcal{C}_2$ denotes the two element chain) are tractable constraint languages for W-MAX CSP$(\cdot)$ this does, for example, currently not imply any complexity results for MAX CSP$(\text{Spmod}_{\mathcal{A}})$ where $\mathcal{A} \in \{\mathcal{C}_2\} \circ \{\mathcal{M}_3\}$ is the lattice in Figure 10.1(a). This should be compared with the results obtained in Chapter 8 which *do*, via Theorem 10.7, imply oracle-pseudo-tractability for $\mathcal{A}$ and hence that MAX CSP$(\text{Spmod}_{\mathcal{A}})$ is in **PO**. On the other hand, the results obtained in Chapter 8 are weaker, compared to the results obtained in [109], in the sense that they only imply a pseudopolynomial-time algorithm for W-MAX CSP$(\text{Spmod}_{\mathcal{M}_k})$ instead of a polynomial time algorithm.

From Theorem 10.7 we get another proof of Lemma 7.6 (which we restate here).

Lemma 7.6 (again)**.** *If* $\mathbf{A}$ *and* $\mathbf{B}$ *are oracle-tractable (oracle-pseudo-tractable, well-characterised) classes of lattices, then so is* $\mathbf{A} \cup \mathbf{B}$*.*

Proof. For any $\mathcal{L} = (L; \sqcap, \sqcup) \in \mathbf{A}$ we have $\mathcal{L} \in \mathbf{A} \circ \mathbf{B}$. To see this, let θ be the full binary relation on L, i.e., $\theta = L^2$. There is only one θ-class, namely L, and $\mathcal{L} \in \mathbf{A}$. Furthermore $\mathcal{L}/\theta$ is the one element lattice and can thus be assumed to be contained in $\mathbf{B}$ (the one element lattice does not change the complexity of the SFM problem). Similarly, if $\mathcal{L} \in \mathbf{B}$, then let θ be the equality relation on $\mathcal{L}$. In this case the θ-classes are one element and $\mathcal{L}/\theta = \mathcal{L} \in \mathbf{B}$. The result now follows from Theorem 10.7. □

The following result follows from a more general theorem proved by Mal'tsev. [114] We give a proof of the result we need here.

Theorem 10.8. *If* $\mathbf{C}$ *is a class of lattices which is closed under taking sublattices, then* $\mathbf{C} \circ \mathbf{C}$ *is closed under taking sublattices.*

Proof. Let $\mathcal{S} = (S; \sqcap, \sqcup)$ be a sublattice of $\mathcal{L} = (L; \sqcap, \sqcup) \in \mathbf{C} \circ \mathbf{C}$. Let θ be a congruence relation on $\mathcal{L}$ so that $\mathcal{L}/\theta \in \mathbf{C}$ and the congruence classes of θ are contained in $\mathbf{C}$. Let ϕ be the homomorphism from $\mathcal{L}$ to $\mathcal{L}/\theta$ associated with θ.

Now, let θ_S be the relation θ restricted to $S \times S$, i.e., $\theta_S = \theta \cap S^2$. The image of S under ϕ is a sublattice of $\mathcal{L}/\theta$, namely $\mathcal{S}/\theta_S$. (For each $x, y \in \phi(S)$ there are $x', y' \in S$ such that $x = \phi(x')$ and $y = \phi(y')$. Furthermore, $x \sqcap y = \phi(x') \sqcap \phi(y') = \phi(x' \sqcap y') \in \phi(S)$. Closure under $\sqcup$ can be shown analogously.) As $\mathcal{L}/\theta \in \mathbf{C}$ and $\mathbf{C}$ is closed under taking sublattices it follows that $\mathcal{S}/\theta_S \in \mathbf{C}$ as well. It remains to prove that every θ_S-class is contained in $\mathbf{C}$. For every $x \in S$ we have that x/θ_S is a sublattice of x/θ. From

- for every $x \in L$, $x/\theta \in \mathbf{C}$; and
- $\mathbf{C}$ is closed under taking sublattices

it now follows that $x/\theta_S \in \mathbf{C}$ for every $x \in S$. □

10.4 Classes of Non-hard Lattices

For a class of finite lattices $\mathbf{C}$ we will be interested in the class of lattices which can be generated from the lattices in $\mathbf{C}$ by repeatedly taking sublattices, homomorphic images and Mal'tsev products. By the results in the previous sections one could guess that if $\mathbf{C}$ is non-hard in some sense, then such an iterated construction gives rise to new non-hard classes of lattices. We will make this intuition more precise in the current section.

As a start we will state a characterisation result for the lattices which can be constructed by such an iterated application of the constructions available to us. To do this we introduce a function f which will be applied in an iterated fashion.

Let $\mathbf{V}$ be the class of all classes of finite lattices. Define the function $f : \mathbf{V} \to \mathbf{V}$ so that for any $\mathbf{X} \in \mathbf{V}$ we have

- $\mathbf{X} \subseteq f(\mathbf{X})$,
- if $\mathcal{L} \in \mathbf{X}$ and $\mathcal{S}$ is a sublattice of $\mathcal{L}$, then $\mathcal{S} \in f(\mathbf{X})$,
- if $\mathcal{L} \in \mathbf{X}$ and $\mathcal{H}$ is a homomorphic image of $\mathcal{L}$, then $\mathcal{H} \in f(\mathbf{X})$,
- $\mathbf{X} \circ \mathbf{X} \subseteq f(\mathbf{X})$,

and no more lattices are contained in $f(\mathbf{X})$. Let $f^1(\mathbf{X}) = f(\mathbf{X})$ and for an integer $i > 1$ let $f^i(\mathbf{X}) = f(f^{i-1}(\mathbf{X}))$.

Lemma 10.9. *For a class of finite lattices* $\mathbf{C}$*,*

$$\bigcup_{i=1}^{\infty} f^i(\mathbf{C}) \tag{10.5}$$

is the smallest class of finite lattices which contains $\mathbf{C}$ *and is closed under taking sublattices, homomorphic images and Mal'tsev products.*

To simplify the notation a bit we will write MPVAR($\mathbf{C}$) instead of (10.5) in the sequel (MPVAR stands for "Mal'tsev pseudovariety").

Proof. Let $\mathbf{D} = \text{MPVAR}(\mathbf{C})$. We claim that $\mathbf{D}$ is a fixed point of f, i.e., $f(\mathbf{D}) = \mathbf{D}$.

Assume, for the sake of contradiction, that this is not the case. Then $\mathbf{D} \subsetneq f(\mathbf{D})$ and there is some lattice $\mathcal{L} \in f(\mathbf{D})$ such that $\mathcal{L} \notin \mathbf{D}$. However, as $\mathcal{L} \in f(\mathbf{D})$ the lattice $\mathcal{L}$ can be constructed from some lattices $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m$ in $\mathbf{D}$ by taking sublattices, homomorphic images or Mal'tsev products. As $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m \in \mathbf{D}$ it follows that there is some integer n such that $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m \in f^n(\mathbf{C})$. But this means that $\mathcal{L} \in f^{n+1}(\mathbf{C})$ and as $f^{n+1}(\mathbf{C}) \subseteq \mathbf{D}$ we have a contradiction.

As $\mathbf{D}$ is a fixed point of f, it follows that $\mathbf{D}$ is closed under taking sublattices, homomorphic images, and Mal'tsev products. Conversely, any class of finite lattices which is closed under taking sublattices, homomorphic

images, and Mal'tsev products, and contains $\mathbf{C}$ is a fixed point of f and contains $\mathbf{D}$. □

We will show that the class of lattices known to be oracle-tractable (oracle-pseudo-tractable, well-characterised) are all of the form MPVAR($\mathbf{C}$) for some class $\mathbf{C}$ (here $\mathbf{C}$ may depend on the non-hardness notion).

Theorem 10.10. *Let* $\mathbf{C}$ *be a class of lattices which is closed under taking sublattices. If* $\mathbf{C}$ *is oracle-tractable (oracle-pseudo-tractable, well-characterised), then so is* MPVAR($\mathbf{C}$).

Proof. We will prove the theorem for the case when $\mathbf{C}$ is oracle-tractable. The two other cases of non-hardness are proved analogously. As $\mathbf{C}$ is closed under taking sublattices it follows from Theorem 10.8 that any lattice obtained as a sublattice from a Mal'tsev product of some lattices in $\mathbf{C}$ can in fact be obtained by a Mal'tsev product of some lattices in $\mathbf{C}$. By Theorem 10.7 it follows that the Mal'tsev product of two oracle-tractable classes of lattices is oracle-tractable.

Furthermore, by Theorem 10.4 any homomorphic image of an oracle-tractable lattice is oracle-tractable. By a general result in universal algebra, if $\mathcal{L}'$ is a sublattice of a homomorphic image of some lattice $\mathcal{L}$, then $\mathcal{L}'$ is a homomorphic image of a sublattice of $\mathcal{L}$. (This result applies to general algebras, not only lattices. See, e.g., [27, Lemma 9.2] for a proof.)

By these two observations it follows that for any class of lattices $\mathbf{C}$ which is closed under taking sublattices the class

$$\{\mathcal{H} \mid \mathcal{L} \in \mathbf{C} \text{ and } \mathcal{H} \text{ is a homomorphic image of } \mathcal{L}\} \cup (\mathbf{C} \circ \mathbf{C})$$

is closed under taking sublattices. Hence, by Lemma 10.9 it follows that for any lattice $\mathcal{L} \in$ MPVAR($\mathbf{C}$) there is a finite sequence of homomorphic image and Mal'tsev product constructions ending with $\mathcal{L}$ and starting with lattices contained in $\mathbf{C}$ (note that we do not need to take sublattices). It follows that MPVAR($\mathbf{C}$) is oracle-tractable. □

To state the non-hardness results for classes of lattices we need to define the classes we are starting from.

Definition 10.11 (Classes of lattices)**.** *We define the following classes of lattices:*

- $\mathbf{D}_{\text{fin}}$ *is the class of all finite distributive lattices, and*
- $\mathbf{M}_{\text{fin}}$ *is the class of all finite modular lattices.*

We are now ready to prove the results we get by combining Theorem 10.10 and the known non-hardness results for SFM.

Theorem 10.12. *The class* MPVAR($\mathbf{D}_{\text{fin}}$) *is oracle-tractable.*

Proof. It is known that $\mathbf{D}_{\text{fin}}$ is oracle-tractable [109, 133]. The result now follows from Theorem 10.10. □

Known complexity	Definition	Class denoted by
oracle-tractable	$\mathrm{MPVAR}(\mathbf{D}_{\mathrm{fin}})$	**O**
oracle-pseudo-tractable	$\mathrm{MPVAR}(\mathbf{D}_{\mathrm{fin}} \cup \{\mathcal{M}_k \mid k \in \mathbb{N}, k \geq 3\})$	**P**
well-characterised	$\mathrm{MPVAR}(\mathbf{M}_{\mathrm{fin}})$	**W**

Table 10.1: Our current knowledge of the tractability of SFM. The class of lattices obtained from $\mathbf{D}_{\mathrm{fin}}$ by iterating homomorphic images and Mal'tsev products was shown to be oracle-tractable in [109]. The other results are from this thesis. As mentioned after Theorem 10.14 it might be possible to enlarge the class of well-characterised lattices by using the results in Section 9.4.3.

Theorem 10.13. *The class*

$$\mathrm{MPVAR}(\mathbf{D}_{\mathrm{fin}} \cup \{\mathcal{M}_k \mid k \in \mathbb{N}, k \geq 3\})$$

is oracle-pseudo-tractable.

Proof. Let $\mathbf{C}$ denote the class $\mathbf{D}_{\mathrm{fin}} \cup \{\mathcal{M}_k \mid k \in \mathbb{N}, k \geq 3\}$. By the known results for $\mathbf{D}_{\mathrm{fin}}$ [133], Corollary 8.31, and Lemma 7.6 it follows that $\mathbf{C}$ is oracle-pseudo-tractable. Note that $\mathbf{C}$ is closed under taking sublattices. The result now follows from Theorem 10.10. □

Theorem 10.14. *The class* $\mathrm{MPVAR}(\mathbf{M}_{\mathrm{fin}})$ *is well-characterised.*

Proof. By Theorem 9.14 and Lemma 7.6 it follows that $\mathbf{M}_{\mathrm{fin}}$ is well-characterised. As a sublattice of a modular lattice is modular the result follows from Theorem 10.10. □

We note that it might be possible to derive a more general theorem by using Theorem 9.15 instead of Theorem 9.14 in the result above. Our current knowledge of the tractability of SFM over finite lattices is summarised in Table 10.1.

It is natural to ask if any of the three classes of lattices in Table 10.1 collapse and if some of them contains or is identical to some well-known class of lattices. It is clear that $\mathbf{O} \subseteq \mathbf{P} \subseteq \mathbf{W}$, which is not surprising. However, note that we do not have an implication which says that if some lattice $\mathcal{L}$ is oracle-pseudo-tractable, then $\mathcal{L}$ is also well-characterised. So the second inclusion, $\mathbf{P} \subseteq \mathbf{W}$, is not obvious a priori.

It is known that the diamonds are not contained in **O**. [109] Hence, as the diamonds are contained in **P** it follows that $\mathbf{O} \subsetneq \mathbf{P}$. It is not known whether $\mathbf{P} = \mathbf{W}$ or not. However, the subspaces of the vector space $\mathbb{Z}_2^3$ (diagrammed in Figure 9.1) is a plausible candidate for a lattice contained in **W**, but not in **P**. We have not been able to prove this, though. A result which we can prove is that **W** does not contain all finite lattices.

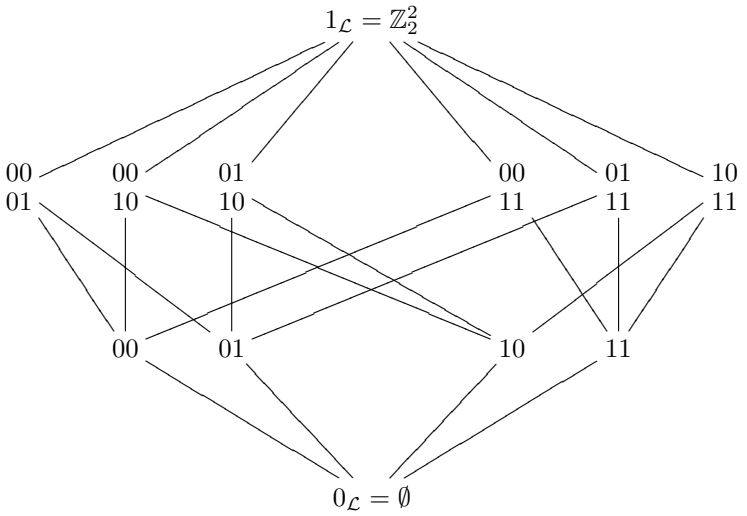


Figure 10.2: The lattice $A(\mathbb{Z}_2; 2)$ of affine subspaces of $\mathbb{Z}_2^2$.

Given a finite field $\mathbb{F}$ and an n-dimensional vector space V over $\mathbb{F}$ an *affine subspace* A of V is a subset $A \subseteq V$ such that if $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m} \in A$, then

$$\sum_{i=1}^{m} a_i \cdot \boldsymbol{x_i} \in A$$

whenever $\sum_{i=1}^{m} a_i = 1$. When the affine subspaces of a vector space are ordered under inclusion they form a lattice. [13, Section I.8] We denote this lattice by $A(\mathbb{F}; n)$.

In [109] it was shown that no diamond is contained in the class generated from $\mathbf{D}_{\mathrm{fin}} \cup \{\mathcal{M}_k \mid k \in \mathbb{N}\}$ by taking homomorphic images and Mal'tsev products. By using the same technique we can show that $A(\mathbb{Z}_2; 2)$ (diagrammed in Figure 10.2) is not contained in $\mathbf{W} = \mathrm{MPVAR}(\mathbf{M}_{\mathrm{fin}})$. It is not hard to see that $A(\mathbb{Z}_2; 2)$ is not modular. Indeed, note that $A(\mathbb{Z}_2; 2)$ has a rank function, i.e., there is a function $\rho : A(\mathbb{Z}_2; 2) \to \mathbb{N}$ which satisfies $\rho(\emptyset) = 0$ and $\rho(x) = \rho(y) + 1$ for all $x, y \in A(\mathbb{Z}_2; 2)$ such that $y \prec x$. Now,

$$\begin{aligned} 4 &= \rho(\{00, 01\}) + \rho(\{10, 11\}) \\ &\neq \rho(\{00, 01\} \sqcup \{10, 11\}) + \rho(\{00, 01\} \sqcap \{10, 11\}) \\ &= 3 \end{aligned}$$

as $\{00, 01\} \sqcup \{10, 11\} = \mathbb{Z}_2^2$ and $\{00, 01\} \sqcap \{10, 11\} = \emptyset$. Hence, ρ is not modular and thus $A(\mathbb{Z}_2; 2)$ is not a modular lattice.

We first state a general lemma which is proved with the same techniques

as was used in [109]. We will then use this lemma to show that $A(\mathbb{Z}_2; 2)$ is not contained in $\mathbf{W}$.

Lemma 10.15. *Let $\mathbf{X}$ be a set of finite lattices and let $\mathcal{L}$ be a lattice such that no lattice in $\mathbf{X}$ has $\mathcal{L}$ as a sublattice. If $\mathcal{L}$ is simple and satisfies the following property:*

> *for every lattice $\mathcal{L}'$, if $\mathcal{L}$ is a homomorphic image of $\mathcal{L}'$, then $\mathcal{L}$ is a sublattice of $\mathcal{L}'$;*

then $\mathcal{L} \notin \mathrm{MPVAR}(\mathbf{X})$.

Proof. Assume, for the sake of contradiction, that $\mathcal{L} \in \mathrm{MPVAR}(\mathbf{X})$. By Lemma 10.9 there is a minimal integer i such that there is a lattice $\mathcal{S} \in f^i(\mathbf{X})$ which has $\mathcal{L}$ as a sublattice. There are now two possibilities, either $\mathcal{S}$ is a homomorphic image of some lattice in $f^{i-1}(\mathbf{X})$ or $\mathcal{S} \in f^{i-1}(\mathbf{X}) \circ f^{i-1}(\mathbf{X})$.

In the first case there is some lattice $\mathcal{H} \in f^{i-1}(\mathbf{X})$ which has $\mathcal{S}$ as a homomorphic image. However, this implies that $\mathcal{L}$ is a homomorphic image of some sublattice of $\mathcal{H}$. (This implication is true for general algebras, not only lattices. [27, Lemma 9.2] We used the same argument in the proof of Theorem 10.10.) By the assumption in the lemma this implies that $\mathcal{L}$ is a sublattice of $\mathcal{H}$, which contradicts the minimality of i.

Hence, we can conclude that $\mathcal{S} \in f^{i-1}(\mathbf{X}) \circ f^{i-1}(\mathbf{X})$. It follows that there is a congruence θ on $\mathcal{S}$ such that $\mathcal{S}/\theta \in f^{i-1}(\mathbf{X})$ and every θ-class is contained in $f^{i-1}(\mathbf{X})$. It is not hard to check that $\theta \cap \mathcal{L}^2$ (that is, θ restricted to $\mathcal{L}$) is a congruence relation on $\mathcal{L}$. As $\mathcal{L}$ is simple the congruence relation $\theta \cap \mathcal{L}^2$ is either the equality relation or the full binary relation on $\mathcal{L}$. In the first case there is a sublattice of $\mathcal{S}/\theta$ which is isomorphic to $\mathcal{L}$ and hence there is some lattice in $f^{i-1}(\mathbf{X})$ which has $\mathcal{L}$ as a sublattice. In the second case some θ-class contains a sublattice isomorphic to $\mathcal{L}$ and hence $f^{i-1}(\mathbf{X})$ contains a lattice which has $\mathcal{L}$ as a sublattice. In either case the minimality of i is contradicted. We conclude that $\mathcal{L} \notin \mathrm{MPVAR}(\mathbf{X})$. □

We will now show that $A(\mathbb{Z}_2; 2)$ is simple and then, in Corollary 10.17, we show that $A(\mathbb{Z}_2; 2) \notin \mathbf{W}$.

Lemma 10.16. *The lattice $A(\mathbb{Z}_2; 2)$ is simple.*

Proof. To simplify the notation somewhat let $\mathcal{L} = A(\mathbb{Z}_2; 2)$. Let ρ be the rank function of $\mathcal{L}$. Assume, for the sake of contradiction, that θ is a non-trivial congruence relation.

First assume that $(x, 0_\mathcal{L}) \in \theta$ for some $x \in \mathcal{L} \setminus \{0_\mathcal{L}, 1_\mathcal{L}\}$. We can assume, without loss of generality, that $\rho(x) = 1$ (as any congruence class is an interval). For any $x \in \mathcal{L}$ such that $\rho(x) = 1$ there are distinct $y_1, y_2, y_3 \in \mathcal{L}$ such that $\rho(y_1) = \rho(y_2) = \rho(y_3) = 2$ and $x \sqcup y_1 = x \sqcup y_2 = x \sqcup y_3 = 1_\mathcal{L}$. As $0_\mathcal{L} \sqcup y_i = y_i$ for all $i \in [3]$ we must have $(y_i, 1_\mathcal{L}) \in \theta$ for all $i \in [3]$. This implies that y_1, y_2, and y_3 are all θ-related to each other. However, $y_1 \sqcap y_2 \sqcap y_3 = 0_\mathcal{L}$ which means that $(0_\mathcal{L}, 1_\mathcal{L}) \in \theta$, contradicting the non-triviality of θ.

Similarly, there cannot be any $x \in \mathcal{L} \setminus \{0_\mathcal{L}, 1_\mathcal{L}\}$ such that $(x, 1_\mathcal{L}) \in \theta$. Hence, there are $x, y \in \mathcal{L}$ such that $\rho(x) = 1, \rho(y) = 2$ and $(x, y) \in \theta$. However, for any such pair there is $z \in \mathcal{L}$ with $\rho(z) = 2$ such that $x \sqcup z = z$ and $y \sqcup z = 1_\mathcal{L}$. As we do not have $(1_\mathcal{L}, z) \in \theta$ this leads to a contradiction. □

Corollary 10.17. *The lattice* $A(\mathbb{Z}_2; 2)$ *is not contained in* $\mathrm{MPVAR}(\mathbf{M}_{\mathrm{fin}})$.

Proof. By Lemma 10.16 $A(\mathbb{Z}_2; 2)$ is simple. By using [92, Theorem 2.47] one can check that if $A(\mathbb{Z}_2; 2)$ is a homomorphic image of some lattice $\mathcal{L}$, then $A(\mathbb{Z}_2; 2)$ is a sublattice of $\mathcal{L}$. The result now follows from Lemma 10.15 together with the observations that $A(\mathbb{Z}_2; 2)$ is not modular and that $\mathbf{M}_{\mathrm{fin}}$ is closed under taking sublattices. □

Let $\mathcal{L}$ denote the lattice of subspaces of $\mathbb{Z}_2^3$ (diagrammed in Figure 9.1). It was mentioned above that $\mathcal{L}$ is a plausible candidate for a lattice contained in $\mathbf{W}$ but not in $\mathbf{P}$. It is clear that $\mathcal{L} \in \mathbf{W}$, as $\mathcal{L}$ is modular. Unfortunately, it is not possible to use the same argument as in Corollary 10.17 to show $\mathcal{L}$ is not contained in $\mathbf{P}$. The problem is that [92, Theorem 2.47] is not applicable, so one cannot conclude that $\mathcal{L}$ is a sublattice of a lattice $\mathcal{L}'$ whenever $\mathcal{L}$ is a homomorphic image of $\mathcal{L}'$. As this is needed in Theorem 10.15 the argument breaks down.

Chapter 11

Hard CSPs Have Hard Gaps

In the remaining two chapters of this part of the thesis we will not study the SFM problem, but instead study the complexity of Max CSP(Γ) directly, for various choices of Γ.

In this chapter we show that if a certain conjecture regarding the complexity of CSP(Γ) is true and CSP(Γ) is **NP**-hard, then it is **NP**-hard to distinguish completely satisfiable instances of Max CSP(Γ) from those where some constant fraction of the constraints can be satisfied (in particular if it follows from known results that CSP(Γ) is **NP**-hard, then there is no ptas for Max CSP(Γ) unless $\mathbf{P} = \mathbf{NP}$).

This result holds even if the number of occurrences of each variable is bounded by a constant. We use our result to answer some open questions.

11.1 Introduction

In this chapter we study the family of all constraint languages Γ such that it is currently known that CSP(Γ) is **NP**-complete. We prove that each constraint language in this family makes Max CSP(Γ) have a hard gap at location 1, even when the number of variable occurrences in an instance is bounded by a sufficiently large constant (depending on Γ), see Theorem 11.9. "Hard gap at location 1" means that it is **NP**-hard to distinguish instances of Max CSP(Γ) in which all constraints are satisfiable from instances where at most an ϵ-fraction of the constraints are satisfiable (for some constant ϵ which depends on Γ). This property immediately implies approximation hardness (in particular, no ptas) for the problem, even when restricted to satisfiable instances (Corollary 11.15). We note that, for the boolean domain and without the bounded occurrence restriction, Theorem 11.9 follows from a result of Khanna et al. [102, Theorem 5.14].

Interestingly, the PCP theorem is equivalent to the fact that, for *some* constraint language Γ over some finite set D, MAX CSP(Γ) has a hard gap at location 1 [8, 52, 140]; clearly, CSP(Γ) cannot be polynomial time solvable in this case. Theorem 11.9 means that MAX CSP(Γ) has a hard gap at location 1 for *any* constraint language such that CSP(Γ) is known to be **NP**-complete. Moreover, if the above mentioned conjecture holds, then MAX CSP(Γ) has a hard gap at location 1 whenever CSP(Γ) is not in **P**. Another equivalent reformulation of the PCP theorem states that the problem MAX 3-SAT has a hard gap at location 1 [8, 140], and our proof consists of a gap preserving reduction from this problem through a version of the algebraic argument from [25].

We also use our main result to give partial answers to two open questions. The first one was posed by Engebretsen et al. [55] and concerns the approximability of a finite group problem while the second was posed by Feder et al. [59] and concerns the hardness of CSP(Γ) with the restriction that each variable occurs at most a constant number of times, under the assumption that CSP(Γ) is **NP**-complete.

As mentioned in Chapter 7 Raghavendra [127] has proved almost tight inapproximability results for MAX CSP(Γ) for every Γ, assuming the UGC. We note that one cannot apply Raghavendra's result to obtain the hardness result we get in this chapter. In particular, the results in [127] does not tell us anything about the complexity of telling satisfiable instances from instances where an ϵ-fraction of the constraints are satisfiable, for any $\epsilon < 1$. Furthermore, our results give explicit methods for characterising the class of constraint languages that are "hard". We also do not need any more assumptions than $\mathbf{P} \neq \mathbf{NP}$ to obtain our results (the UGC is used in [127]).

Here is an overview of this chapter: in Section 11.2 we define some concepts we need. In Section 11.3 we prove two lemmas which will be useful later on. In Section 11.4 we define some concepts from universal algebra and connect them with the constraint satisfaction problem. Finally, in Section 11.5 we prove the main result and show how it can be used to answer the two open problems.

11.2 Preliminaries

Throughout this chapter, MAX CSP(Γ)-k will denote the problem MAX CSP(Γ) restricted to instances where the number of occurrences of each variable is bounded by k. For our hardness results we will write that MAX CSP(Γ)-B is hard (in some sense) with the meaning that there is a k such that MAX CSP(Γ)-k is hard in this sense.

Definition 11.1 (Hard to approximate)**.** *We say that a problem* Π *is* hard to approximate *if there exists a constant* $c > 1$ *such that* Π *is* **NP***-hard to approximate within* c *(so the existence of a polynomial-time approximation algorithm for* Π *with performance ratio* c *implies* $\mathbf{P} = \mathbf{NP}$*).*

The following notion has been defined in a more general setting by Petrank [124].

Definition 11.2 (Hard gap at location α)**.** MAX CSP(Γ) *has a* hard gap at location $\alpha \leq 1$ *if there exists a constant* $\epsilon < \alpha$ *and a polynomial-time reduction from an* **NP***-complete problem* Π *to* MAX CSP(Γ) *such that,*

- YES *instances of* Π *are mapped to instances* $I = (V, C)$ *such that* $\text{OPT}(I) \geq \alpha|C|$*, and*
- NO *instances of* Π *are mapped to instances* $I = (V, C)$ *such that* $\text{OPT}(I) \leq \epsilon|C|$.

Note that if a problem Π has a hard gap at location α (for any α) then Π is hard to approximate. This simple observation has been used to prove inapproximability results for a large number of optimisation problems. See, e.g., [7, 9, 140] for surveys on inapproximability results and the related PCP theory.

11.3 Proof Techniques

In this section we show two lemmas which relates hardness at gap location 1 to pp-definitions and cores.

Lemma 11.3. *If a finite constraint language* Γ *can pp-define a relation* R *and* MAX CSP($\Gamma \cup \{R\}$)*-*k *has a hard gap at location 1, then* MAX CSP(Γ)*-*k' *has a hard gap at location 1 for some integer* k'*.*

Proof. Let N be the minimum number of relations that are needed in a pp-definition of R using relations from Γ.

Given an instance $I = (V, C)$ of MAX CSP($\Gamma \cup \{R\}$)-k, we construct an instance $I' = (V', C')$ of MAX CSP(Γ)-k' (where k' will be specified below). We will use the set V'' to store auxiliary variables during the reduction so we initially let V'' be the empty set. For a constraint $c = (Q, \boldsymbol{s}) \in C$, there are two cases to consider:

1. If $Q \neq R$, then add N copies of c to C'.
2. If $Q = R$, then add the implementation of R to C' where any auxiliary variables in the implementation are replaced with fresh variables which are added to V''.

Finally, let $V' = V \cup V''$. It is clear that there exists an integer k', independent of I, such that I' is an instance of MAX CSP(Γ')-k'.

If all constraints are simultaneously satisfiable in I, then all constraints in I' are also simultaneously satisfiable. On the other hand, if $\text{OPT}(I) \leq \epsilon|C|$ then

$$\begin{aligned}\text{OPT}(I') &\leq \epsilon N|C| + (1-\epsilon)(N-1)|C| \\ &= (\epsilon + (1-\epsilon)(1-1/N))\,|C'|.\end{aligned}$$

The inequality holds because each constraint in I introduces a group of N constraints in I' and, as $\text{OPT}(I) \leq \epsilon|C|$, at most $\epsilon|C|$ such groups are completely satisfied. In all other groups (there are $(1-\epsilon)|C|$ such groups) at least one constraint is not satisfied. Furthermore, $\epsilon+(1-\epsilon)(1-1/N) < 1$ so we can conclude that MAX CSP(Γ)-k' has a hard gap at location 1. □

The following simple lemma connects cores with hardness at gap location 1.

Lemma 11.4. *If Γ' is the core of Γ, then, for any k,* MAX CSP(Γ')-k *has a hard gap at location 1 if and only if* MAX CSP(Γ)-k *has a hard gap at location 1.*

Proof. Let π be the retraction of Γ such that $\Gamma' = \{\pi(R) \mid R \in \Gamma\}$, where $\pi(R) = \{\pi(\boldsymbol{t}) \mid \boldsymbol{t} \in R\}$. Given an instance $I = (V, C)$ of MAX CSP(Γ)-k, we construct an instance $I' = (V, C')$ of MAX CSP(Γ')-k by replacing each constraint $(R, \boldsymbol{s}) \in C$ by $(\pi(R), \boldsymbol{s})$.

From a solution s to I, we construct a solution s' to I' such that $s'(x) = \pi(s(x))$. Let $(R, \boldsymbol{s}) \in C$ be a constraint which is satisfied by s. Then, there is a tuple $\boldsymbol{x} \in R$ such that $s(\boldsymbol{s}) = \boldsymbol{x}$ so $\pi(\boldsymbol{x}) \in \pi(R)$ and $s'(\boldsymbol{s}) = \pi(s(\boldsymbol{s})) = \pi(\boldsymbol{x}) \in \pi(R)$. Conversely, if $(\pi(R), \boldsymbol{s})$ is a constraint in I' which is satisfied by s', then there is a tuple $\boldsymbol{x} \in R$ such that $s'(\boldsymbol{s}) = \pi(s(\boldsymbol{s})) = \pi(\boldsymbol{x}) \in \pi(R)$, and $s(\boldsymbol{s}) = \boldsymbol{x} \in R$. We conclude that $m(I, s) = m(I', s')$.

It is not hard to see that we can do this reduction in the other direction too, i.e., given an instance $I' = (V', C')$ of MAX CSP(Γ')-k, we construct an instance I of MAX CSP(Γ)-k by replacing each constraint $(\pi(R), \boldsymbol{s}) \in C'$ by $(R, \boldsymbol{s})$. By the same argument as above, this direction of the equivalence follows, and we conclude that the lemma is valid. □

An analogous result holds for the CSP problem, i.e., if Γ' is the core of Γ, then CSP(Γ) is in **P** (**NP**-complete) if and only if CSP(Γ') is in **P** (**NP**-complete); see [89] for a proof.

11.4 Constraint Satisfaction and Algebra

In this section we will present some definitions and basic results we need from universal algebra, in addition to the concepts of clones, co-clones and polymorphisms described in Section 2.2. For a more thorough treatment of universal algebra in general we refer the reader to [27, 38]. The articles [25, 37] contain presentations of the relationship between universal algebra and constraint satisfaction problems. The three definitions below closely follow the presentation in [25].

Definition 11.5 (Finite algebra)**.** *A* finite algebra *is a pair $\mathcal{A} = (A; F)$ where A is a finite non-empty set and F is a set of finitary operations on A.*

We will only make use of finite algebras so we will write *algebra* instead of *finite algebra*. An algebra is said to be *non-trivial* if it has more than

one element. In Chapter 10 we defined homomorphisms and subalgebras for lattices. We will now generalise these definitions to algebras.

Definition 11.6 (Homomorphism of algebras)**.** *Given two algebras* $\mathcal{A} = (A; F_A)$ *and* $\mathcal{B} = (B; F_B)$ *such that* $F_A = \{f_i^A \mid i \in I\}$, $F_B = \{f_i^B \mid i \in I\}$ *and both* f_i^A *and* f_i^B *are* n_i*-ary for all* $i \in I$*, then* $\phi : A \to B$ *is said to be an* homomorphism *from* $\mathcal{A}$ *to* $\mathcal{B}$ *if*

$$\phi(f_i^A(a_1, a_2, \ldots, a_{n_i})) = f_i^B(\phi(a_1), \phi(a_2), \ldots, \phi(a_{n_i}))$$

for all $i \in I$ *and* $a_1, a_2, \ldots, a_{n_i} \in A$*. If* ϕ *is surjective, then* $\mathcal{B}$ *is a* homomorphic image *of* $\mathcal{A}$*.*

Given a homomorphism ϕ mapping $\mathcal{A} = (A; F_A)$ to $\mathcal{B} = (B; F_B)$, we can construct an equivalence relation θ on A as $\theta = \{(x, y) \mid \phi(x) = \phi(y)\}$. The relation θ is said to be a *congruence relation* of $\mathcal{A}$. We can now construct the *quotient algebra* $\mathcal{A}/\theta = (A/\theta; F_A/\theta)$. Here, $A/\theta = \{x/\theta \mid x \in A\}$ and x/θ is the equivalence class containing x. Furthermore, $F_A/\theta = \{f/\theta \mid f \in F_A\}$ and f/θ is defined such that $f/\theta(x_1/\theta, x_2/\theta, \ldots, x_n/\theta) = f(x_1, x_2, \ldots, x_n)/\theta$.

Definition 11.7 (Subalgebra)**.** *Let* $\mathcal{A} = (A; F_A)$ *be an algebra and* $B \subseteq A$*. If for each* $f \in F_A$ *and any* $b_1, b_2, \ldots, b_n \in B$*, we have* $f(b_1, b_2, \ldots, b_n) \in B$*, then* $\mathcal{B} = (B; F_A|_B)$ *is a* subalgebra *of* $\mathcal{A}$*.*

The operations in $\mathrm{Pol}(\mathrm{Inv}(F_A))$ are the *term operations* of $\mathcal{A}$. If all term operations are surjective, then the algebra is said to be *surjective*. Note that $\mathrm{Inv}(F_A)$ is a core if and only if $\mathcal{A}$ is surjective [25, 89]. If F consist of all the idempotent term operations of $\mathcal{A}$, then the algebra $(A; F)$ is called the *full idempotent reduct of* $\mathcal{A}$, and we will denote this algebra by $\mathcal{A}^c$. Given a set of relations Γ over the domain D we say that the algebra $\mathcal{A}_\Gamma = (D; \mathrm{Pol}(\Gamma))$ is *associated* with Γ. An algebra $\mathcal{B}$ is said to be a *factor* of the algebra $\mathcal{A}$ if $\mathcal{B}$ is a homomorphic image of a subalgebra of $\mathcal{A}$. A *non-trivial factor* is a factor which is a non-trivial algebra, i.e., it has at least two elements.

We continue by describing some connections between constraint satisfaction problems and universal algebra. The following theorem concerns the hardness of CSP for certain constraint languages.

Theorem 11.8 ([25])**.** *Let* Γ *be a core constraint language. If* $\mathcal{A}_\Gamma^c$ *has a non-trivial factor whose term operations are only projections, then* $\mathrm{CSP}(\Gamma)$ *is* **NP***-hard.*

The algebraic CSP conjecture [25] states that, for all other core languages Γ, the problem $\mathrm{CSP}(\Gamma)$ is tractable. (This conjecture is stronger than Conjecture 1.7 as the latter only says that $\mathrm{CSP}(\Gamma)$ is either in **P** or is **NP**-complete, it does not say anything about *when* the two cases occur.)

The main result of this chapter is the following theorem which states that Max $\mathrm{CSP}(\Gamma)$-B has a hard gap at location 1 whenever the condition which makes $\mathrm{CSP}(\Gamma)$ hard in Theorem 11.8 is satisfied.

Theorem 11.9. *Let Γ be a core constraint language. If $\mathcal{A}_\Gamma^c$ has a non-trivial factor whose term operations are only projections, then* Max CSP(Γ)-B *has a hard gap at location 1.*

The proof of this result can be found in Section 11.5. Note that if the algebraic CSP conjecture is true then Theorem 11.9 describes *all* constraint languages Γ for which Max CSP(Γ) has a hard gap at location 1 because, obviously, Γ cannot have this property when CSP(Γ) is tractable.

11.5 Proofs

In this section, we prove the main result of this chapter, Theorem 11.9. We also give two applications of this theorem. Let $3SAT_0$ denote the relation $\{0,1\}^3 \setminus \{(0,0,0)\}$. We also introduce three slight variations of $3SAT_0$, let $3SAT_1 = \{0,1\}^3 \setminus \{(1,0,0)\}$, $3SAT_2 = \{0,1\}^3 \setminus \{(1,1,0)\}$, and $3SAT_3 = \{0,1\}^3 \setminus \{(1,1,1)\}$. To simplify the notation we let $\Gamma_{3SAT} = \{3SAT_0, 3SAT_1, 3SAT_2, 3SAT_3\}$. It is not hard to see that the problem Max CSP(Γ_{3SAT}) is precisely Max 3Sat. It is well-known that this problem, even when restricted to instances in which each variable occurs at most a constant number of times, has a hard gap at location 1, see e.g., [140, Theorem 7]. We state this as a lemma.

Lemma 11.10 ([140])**.** Max CSP(Γ_{3SAT})-B *has a hard gap at location 1.*

To prove Theorem 11.9 we will utilise *expander graphs*.

Definition 11.11 (Expander graph)**.** *A d-regular graph $G = (V, E)$ is an* expander graph *if, for any $S \subseteq V$, the number of edges between S and $V \setminus S$ is at least* $\min(|S|, |V \setminus S|)$.

Expander graphs are frequently used for proving properties of Max CSP, cf. [46, 123]. Typically, they are used for bounding the number of variable occurrences. A concrete construction of expander graphs has been provided by Lubotzky et al. [113].

Theorem 11.12. *A polynomial-time algorithm T and a fixed integer N exist such that, for any $k > N$, $T(k)$ produces a 14-regular expander graph with $k(1 + o(1))$ vertices.*

There are four basic ingredients in the proof of Theorem 11.9. The first three are Lemma 11.3, Lemma 11.10, and the use of expander graphs to bound the number of variable occurrences. We also use an alternative characterisation (Lemma 11.13) of constraint languages satisfying the conditions of the theorem. This is a slight modification of a part of the proof of Proposition 7.9 in [25]. The implication below is in fact an equivalence and we refer the reader to [25] for the details. Given a function $f : D^n \to D$, and a relation $R \in R_D$, the *full preimage* of R under f, denoted by $f^{-1}(R)$, is the relation $\{\boldsymbol{x} \in D^n \mid f(\boldsymbol{x}) \in R\}$ (as usual, $f(\boldsymbol{x})$ denotes that f should be

applied componentwise to $\boldsymbol{x}$). For any $a \in D$, we denote the unary constant relation containing only a by c_a, i.e., $c_a = \{(a)\}$. Let C_D denote the set of all constant relations over D, that is, $C_D = \{c_a \mid a \in D\}$.

Lemma 11.13. *Let Γ be a core constraint language. If the algebra $\mathcal{A}^c_\Gamma$ has a non-trivial factor whose term operations are only projections, then there is a subset B of D and a surjective mapping $\phi : B \to \{0,1\}$ such that the relational clone $\langle \Gamma \cup C_D \rangle$ contains the relations $\phi^{-1}(3SAT_0)$, $\phi^{-1}(3SAT_1)$, $\phi^{-1}(3SAT_2)$, and $\phi^{-1}(3SAT_3)\}$.*

Proof. Let $\mathcal{A}'$ be the subalgebra of $\mathcal{A}^c_\Gamma$ such that there is a surjective homomorphism ϕ from $\mathcal{A}'$ to a non-trivial algebra $\mathcal{B}$ whose term operations are only projections. We can assume, without loss of generality, that the set $\{0,1\}$ is contained in the universe of $\mathcal{B}$. It is easy to see that any relation is invariant under any projection. Since $\mathcal{B}$ only has projections as term operations, the four relations $3SAT_0$, $3SAT_1$, $3SAT_2$ and $3SAT_3$ are invariant under the term operations of $\mathcal{B}$. It is known (see [25]) that the full preimages of those relations under ϕ are invariant under the term operations of $\mathcal{A}'$ and therefore they are also invariant under the term operations of $\mathcal{A}^c_\Gamma$. By the observation that $\mathcal{A}^c_\Gamma = \mathcal{A}_{\Gamma \cup C_D}$ and Theorem 2.8, this implies $\{\phi^{-1}(3SAT_0)$, $\phi^{-1}(3SAT_1)$, $\phi^{-1}(3SAT_2)$, $\phi^{-1}(3SAT_3)\} \subseteq \langle \Gamma \cup C_D \rangle$. □

We are now ready to present the proof of Theorem 11.9. Let S be a permutation group on the set X. An *orbit* of S is a subset Ω of X such that $\Omega = \{g(x) \mid g \in S\}$ for some $x \in X$.

Proof. By Lemma 11.3, in order to prove the theorem, it suffices to find a finite set $\Gamma' \subseteq \langle \Gamma \rangle$ such that MAX CSP(Γ')-B has a hard gap at location 1.

Since Γ is a core, its unary polymorphisms form a permutation group S on D. We can without loss of generality assume that $D = \{1, \ldots, p\}$. It is known (see Proposition 1.3 of [138]) and possible to check with Theorem 2.8 that Γ can pp-define the following relation: $R_S = \{(g(1), \ldots, g(p)) \mid g \in S\}$. Then it can also pp-define the relations EQ_i for $1 \leq i \leq p$ where EQ_i is the restriction of the equality relation on D to the orbit in S which contains i. We have

$$\begin{aligned} EQ_i(x,y) \iff \exists z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_p : \\ R_S(z_1, \ldots, z_{i-1}, x, z_{i+1}, \ldots, z_p) \wedge \\ R_S(z_1, \ldots, z_{i-1}, y, z_{i+1}, \ldots, z_p). \end{aligned}$$

By Lemma 11.13, there exists a subset (in fact, a subalgebra) B of D and a surjective mapping $\phi : B \to \{0,1\}$ such that the relational clone $\langle \Gamma \cup C_D \rangle$ contains $\phi^{-1}(\Gamma_{3SAT}) = \{\phi^{-1}(R) \mid R \in \Gamma_{3SAT}\}$. For $0 \leq i \leq 3$, let R_i be the full preimage of $3SAT_i$ under ϕ. Since $R_i \in \langle \Gamma \cup C_D \rangle$, we can show that there exists a $(p+3)$-ary relation R'_i in $\langle \Gamma \rangle$ such that

$$R_i = \{(x,y,z) \mid (1,2,\ldots,p,x,y,z) \in R'_i\}.$$

Indeed, since $R_i \in \langle \Gamma \cup C_D \rangle$, R_i can be defined by the pp-formula

$$R_i(x,y,z) \iff \exists \boldsymbol{t} : \psi(\boldsymbol{t},x,y,z)$$

(here $\boldsymbol{t}$ denotes a tuple of variables) where ψ is a conjunction of atomic formulae involving predicates from $\Gamma \cup C_D$ and variables from $\boldsymbol{t}$ and $\{x,y,z\}$. Note that, in ψ, no predicate from C_D is applied to one of $\{x,y,z\}$ because these variables can take more than one value in R_i. We can without loss of generality assume that every predicate from C_D appears in ψ exactly once. Indeed, if such a predicate appears more than once, then we can identify all variables to which it is applied, and if it does not appear at all then we can add a new variable to $\boldsymbol{t}$ and apply this predicate to it. Now assume without loss of generality that the predicate c_i, $1 \leq i \leq p$, is applied to the variable t_i in ψ, and $\psi = \psi_1 \wedge \psi_2$ where $\psi_1 = \bigwedge_{i=1}^{p} c_i(t_i)$ and ψ_2 contains only predicates from $\Gamma \setminus C_D$. Let $\boldsymbol{t'}$ be the list of variables obtained from $\boldsymbol{t}$ by removing $t_1,\dots,t_p$. It now is easy to check that the $(p+3)$-ary relation R_i' defined by the pp-formula $\exists \boldsymbol{t'} : \psi_2(\boldsymbol{t'},x,y,z)$ has the required property.

Choose R_i' to be an inclusion-wise minimal relation in $\langle \Gamma \rangle$ such that

$$R_i = \{(x,y,z) \mid (1,2,\dots,p,x,y,z) \in R_i'\}$$

and let $\Gamma' = \{R_i' \mid 0 \leq i \leq 3\} \cup \{EQ_1,\dots,EQ_p\}$. Note that we have $\Gamma' \subseteq \langle \Gamma \rangle$.

We will need a more concrete description of R_i', so we now show that

$$R_i' = \{(g(1),g(2),\dots,g(p),g(x),g(y),g(z)) \mid g \in S, (x,y,z) \in R_i\}. \quad (11.1)$$

The set on the right-hand side of the above equality must be contained in R_i' because R_i' is invariant under all operations in S. On the other hand, if a tuple $\boldsymbol{b} = (b_1,\dots,b_p,d,e,f)$ belongs to R_i', then there is a permutation $g \in S$ such that $(b_1,\dots,b_p) = (g(1),\dots,g(p))$ (otherwise, the intersection of this relation with $R_S \times D^3 \in \langle \Gamma \rangle$ would give a smaller relation with the required property). Now note that the tuple $(1,\dots,p,g^{-1}(d),g^{-1}(e),g^{-1}(f))$ also belongs to R_i' implying, by the choice of R_i', that $(g^{-1}(d),g^{-1}(e),g^{-1}(f)) \in R_i$. Therefore, the relation R_i' is indeed as described above.

By Lemma 11.10, there is l such that MAX CSP(Γ_{3SAT})-l has a hard gap at location 1. By Lemma 11.4, MAX CSP($\phi^{-1}(\Gamma_{3SAT})$)-l has the same property (because Γ_{3SAT} is the core of $\phi^{-1}(\Gamma_{3SAT})$). To complete the proof, we will now reduce MAX CSP($\phi^{-1}(\Gamma_{3SAT})$)-l to MAX CSP(Γ')-l' where $l' = \max\{14p+1, l\}$ (recall that $p = |D|$ is a constant). Take an arbitrary instance $I = (V,C)$ of MAX CSP($\phi^{-1}(\Gamma_{3SAT})$)-l, and build an instance $I' = (V',C')$ of MAX CSP(Γ') as follows: introduce new variables $u_1,\dots,u_p$, and replace each constraint $R_i(x,y,z)$ in I by $R_i'(u_1,\dots,u_p,x,y,z)$. Note that every variable, except the u_i's, in I' appears at most l times. We will now use expander graphs to construct an instance $I'' = (V' \cup V'', C*)$ of MAX CSP(Γ')-l', where V'' and $C*$ are specified below.

Let q be the number of constraints in I and let $q' = \max\{N,q\}$, where N is the constant in Theorem 11.12. Let $G = (W,E)$ be an expander

graph (constructed in polynomial time by the algorithm in Theorem 11.12 with input q') such that $W = \{w_1, w_2, \ldots, w_m\}$ and $m \geq q$. The expander graph $T(q')$ has $q'(1 + o(1))$ vertices. Hence, there is a constant α such that G has at most αq vertices. For each $1 \leq j \leq p$, we introduce m fresh variables $w_1^j, w_2^j, \ldots, w_m^j$ into V''. For each edge $\{w_i, w_k\} \in E$ and $1 \leq j \leq p$, introduce p copies of the constraint $EQ_j(w_i^j, w_k^j)$ into C''. Let $C_1, C_2, \ldots, C_q$ be an enumeration of the constraints in C'. Replace u_j by w_i^j in C_i for all $1 \leq i \leq q$. Finally, let C^* be the union of the (modified) constraints in C' and the equality constraints in C''. It is clear that each variable occurs in I'' at most $l' = \max\{14p+1, l\}$ times (as G is 14-regular).

Clearly, a solution s to I satisfying all constraints can be extended to a solution to I'', also satisfying all constraints, by setting $s(w_i^j) = j$ for all $1 \leq i \leq m$ and all $1 \leq j \leq p$.

On the other hand, if $m(I, s) \leq \epsilon|C|$ for some $\epsilon < 1$, then let s' be an optimal solution to I''. We will prove that there is a constant $\epsilon' < 1$ (which depends on ϵ but not on I) such that $m(I'', s') \leq \epsilon'|C^*|$.

We first prove that, for each $1 \leq j \leq p$, we can assume that all variables in $W^j = \{w_1^j, w_2^j, \ldots, w_m^j\}$ have been assigned the same value by s' and that all constraints in C'' are satisfied by s'. We show that given a solution s' to I'', we can construct another solution s_2 such that $m(I'', s_2) \geq m(I'', s')$ and s_2 satisfies all constraints in C''.

Let $a^j \in D$ be a value that at least m/p of the variables in W^j have been assigned by s'. We construct the solution s_2 as follows: $s_2(w_i^j) = a^j$ for all i and j, and $s_2(x) = s'(x)$ for all other variables.

If there is some j such that $X = \{x \in W^j \mid s'(x) \neq a^j\}$ is non-empty, then, since G is an expander graph, there are at least $p \cdot \min(|X|, |W^j \setminus X|)$ constraints in C'' which are not satisfied by s'. Note that by the choice of X, we have $|W^j \setminus X| \geq m/p$ which implies $p \cdot \min(|X|, |W^j \setminus X|) \geq |X|$. By changing the value of the variables in X, we will make at most $|X|$ non-equality constraints in C^* unsatisfied because each of the variables in W^j occurs in at most one non-equality constraint in C^*. In other words, when the value of the variables in X are changed we gain at least $|X|$ in the measure as some of the equality constraints in C'' will become satisfied, furthermore we lose at most $|X|$ by making at most $|X|$ constraints in C' unsatisfied. We conclude that $m(I', s_2) \geq m(I', s')$. Thus, we may assume that all equality constraints in C'' are satisfied by s'.

We will now show that we can assume that $s'(w_1^1)$, $s'(w_1^2)$, ..., $s'(w_1^p)$ are distinct (and hence, as s' is equal on the W^j's it follows that $s'(w_i^1)$, ..., $s'(w_i^p)$ are distinct for $1 \leq i \leq m$ as well). If $s'(w_1^1)$, $s'(w_1^2)$, ..., $s'(w_1^p)$ are not distinct, then it follows by (11.1) that no constraint in C' is satisfied by s'. Hence, we can construct a new assignment $s_3 : V' \cup V'' \to D$ such that $s_3(w_i^j) = j$ for all $i \in [m]$ and $j \in [p]$ and $s_3(x) = s'(x)$ for any $x \in V'$. It follows that $m(I'', s_3) \geq m(I'', s')$. Thus, we may assume that there is a permutation g on $[p]$ such that $s'(w_i^j) = g(j)$.

By using (11.1) again we can actually assume that $g \in S$. It follows that

$m(I'', g^{-1} \circ s') \geq m(I'', s')$ and hence we can assume that $s'(w_i^j) = j$ for $i \in [m]$ and $j \in [p]$.

Since the expander graph G is 14-regular and has at most αq vertices, it has at most $\frac{14}{2}\alpha q$ edges. Hence, the number of equality constraints in C'' is at most $7\alpha qp$, and $|C''|/|C'| \leq 7\alpha p$. We can now bound $m(I'', s_2)$ as follows:

$$\begin{aligned} m(I'', s_2) &\leq \text{OPT}(I') + |C''| \\ &\leq \frac{\epsilon|C'| + |C''|}{|C'| + |C''|}(|C'| + |C''|) \\ &\leq \frac{\epsilon + 7\alpha p}{1 + 7\alpha p}(|C'| + |C''|). \end{aligned}$$

Here the first inequality follows from (11.1), $s'(w_i^j) = j$ for $i \in [m], j \in [p]$, and that $s'|_{V'}$ is an assignment to I'. The last inequality follows as $\frac{\epsilon|C'|+|C''|}{|C'|+|C''|}$ is maximised when $|C''|$ attains its maximum value which is $7\alpha qp$. Since $|C^*| = |C'| + |C''|$, it remains to set $\epsilon' = \frac{\epsilon+7\alpha p}{1+7\alpha p}$. □

We finish this section by using Theorem 11.9 to answer, at least partially, two open questions. The first one concerns the complexity of CSP(Γ)-B. In particular, the following conjecture has been made by Feder et al. [59].

Conjecture 11.14. *For any fixed* Γ *such that* CSP(Γ) *is* **NP***-complete there is an integer* k *such that* CSP(Γ)-k *is* **NP***-complete.*

Under the assumption that the algebraic CSP conjecture (that all problems CSP(Γ) not covered by Theorem 11.8 are tractable) holds, an affirmative answer follows immediately from Theorem 11.9. So for all constraint languages Γ such that CSP(Γ) is currently known to be **NP**-complete it is also the case that CSP(Γ)-B is **NP**-complete.

The second result concerns the approximability of equations over non-abelian groups. Petrank [124] has noted that hardness at gap location 1 implies the following: suppose that we restrict ourselves to instances of Max CSP(Γ) such that there exist solutions that satisfy all constraints, i.e. we concentrate on *satisfiable* instances. Then, there exists a constant $c > 1$ (depending on Γ) such that no polynomial-time algorithm can approximate this problem within c. We get the following result for satisfiable instances:

Corollary 11.15. *Let* Γ *be a core constraint language and let* $\mathcal{A}$ *be the algebra associated with* Γ*. Assume there is a factor* $\mathcal{B}$ *of* $\mathcal{A}^c$ *such that* $\mathcal{B}$ *only have projections as term operations. Then, there exists a constant* $c > 1$ *such that* Max CSP(Γ)-B *restricted to satisfiable instances cannot be approximated within* c *in polynomial time.*

We will now use this observation for studying a problem concerning groups. Let $\mathcal{G} = (G, \cdot)$ denote a finite group with identity element 1_G. An *equation* over a set of variables V is an expression of the form $w_1 \cdot \ldots \cdot w_k = 1_G$, where w_i (for $1 \leq i \leq k$) is either a variable, an inverted variable, or a group constant. Engebretsen et al. [55] have studied the following problem:

Definition 11.16 (Max $\mathrm{Eq}_{\mathcal{G}}$)**.** Max $\mathrm{Eq}_{\mathcal{G}}$*, where* $\mathcal{G}$ *is a finite group, is the optimisation problem with*

Instance: *A tuple* (V, E) *where* V *is a set of variables and* E *is a multiset of equations over* V*.*

Solution: *An assignment* $s : V \to G$ *to the variables.*

Measure: *Number of equations in* E *which are satisfied by* s*.*

In Chapter 4 we used inapproximability results for this problem for *abelian* groups (under the name Max Ek-Lin-G). We will need a certain restriction of the problem defined above. The problem Max $\mathrm{Eq}^1_{\mathcal{G}}[3]$ is the same as Max $\mathrm{Eq}_{\mathcal{G}}$ except for the additional restrictions that each equation contains exactly three variables and no equation contains the same variable more than once. Engebretsen et al.'s main result was the following inapproximability result:

Theorem 11.17 (Theorem 1 in [55])**.** *For any finite group* $\mathcal{G}$ *and constant* $\epsilon > 0$*, it is* **NP***-hard to approximate* Max $\mathrm{Eq}^1_{\mathcal{G}}[3]$ *within* $|G| - \epsilon$*.*

Engebretsen et al. left the approximability of Max $\mathrm{Eq}^1_{\mathcal{G}}[3]$ for satisfiable instances as an open question. We will give a partial answer to the approximability of satisfiable instances of Max $\mathrm{Eq}_{\mathcal{G}}$.

It is not hard to see that for any integer k, the equations with at most k variables over a finite group can be viewed as a constraint language. For a group $\mathcal{G}$, we denote the constraint language which corresponds to equations with at most three variables by $\Gamma_{\mathcal{G}}$. Hence, for any finite group $\mathcal{G}$, the problem Max $\mathrm{CSP}(\Gamma_{\mathcal{G}})$ is no harder than Max $\mathrm{Eq}_{\mathcal{G}}$.

Goldmann and Russell [69] have shown that $\mathrm{CSP}(\Gamma_{\mathcal{G}})$ is **NP**-hard for every finite non-abelian group $\mathcal{G}$. This result was extended to more general algebras by Larose and Zádori [111]. They also showed that for any non-abelian group $\mathcal{G}$, the algebra $\mathcal{A} = (G; \mathrm{Pol}(\Gamma_{\mathcal{G}}))$ has a non-trivial factor $\mathcal{B}$ such that $\mathcal{B}$ only has projections as term operations. We now combine Larose and Zádori's result with Theorem 11.9:

Corollary 11.18. *For any finite non-abelian group* $\mathcal{G}$*,* Max $\mathrm{Eq}_{\mathcal{G}}$ *has a hard gap at location 1.*

Thus, there is a constant c such that no polynomial-time algorithm can approximate satisfiable instances of Max $\mathrm{Eq}_{\mathcal{G}}$ better than c. There also exists a constant k (depending on the group $\mathcal{G}$) such that the result holds for instances with variable occurrence bounded by k.

Chapter 12

Single Relation Max CSP

In this chapter we show that for any relation R, Max CSP($\{R\}$) is either trivial (one can satisfy every constraint without looking at the instance) or admits no PTAS (unless $\mathbf{P} = \mathbf{NP}$). This result holds even if the number of occurrences of each variable is bounded by a constant. To prove the result we make use of the main result from Chapter 11. We also give some applications of the result.

12.1 Introduction

Max CSP(Γ) when Γ consists of a single relation contains some of the best-studied examples of Max CSP such as Max Cut and Max DiCut. It was proved in [94] that, for any non-empty relation R, the problem Max CSP($\{R\}$) is either trivial (i.e., mapping all variables in any instance to the same fixed value always satisfies all constraints) or **NP**-hard. We strengthen this result by proving approximation hardness (and hence the non-existence of PTAS) instead of **NP**-hardness (see Theorem 12.10), even with a bound on the number of variable occurrences.

We note that the status of the analogous problem for CSP is very different. A full complexity classification of single-relation CSP is not known. In fact, Feder and Vardi [60] have proved that by providing such a classification, one has also classified the CSP problem for *all* constraint languages thus resolving the Feder-Vardi dichotomy conjecture (Conjecture 1.7). No result of this kind is known for Max CSP.

For some Boolean Max CSP problems, e.g., for Max Cut, a stronger version of Theorem 12.10 is known (see, e.g., [81]). By "stronger" we mean that the results in [81] give better bounds on the performance ratio of any polynomial-time approximation algorithm under the assumption that $\mathbf{P} \neq \mathbf{NP}$. If one is willing to assume the UGC, then, as mentioned in Chapter 7, Raghavendra has proved almost tight inapproximability results for Max CSP(Γ) for every Γ. [127]

Here is an overview of this chapter: in Section 12.2 we define some concepts we need. In Section 12.3 we describe the proof techniques we will use. Section 12.4 contains the proof of the main result. In Section 12.5 we use the main result to generalise some results from [107, 108].

12.2 Preliminaries

We will use the notations MAX CSP(Γ)-k and MAX CSP(Γ)-B in the same way as they were used in Chapter 11. We will also use the notion of "hard to approximate" as defined in Definition 11.1. The following result is well-known (see, e.g., [35, Proposition 2.3]).

Lemma 12.1. *Let D be a finite set. For every constraint language $\Gamma \subseteq R_D$,* MAX CSP(Γ) *belongs to* **APX**. *Moreover, if a is the maximum arity of any relation in Γ, then there is a polynomial time approximation algorithm with performance ratio $|D|^a$*

The algorithm in Lemma 12.1 simply assigns values uniformly at random to the variables. One can prove that this method achieves the performance ratio $|D|^a$. We will use the following known result several times.

Lemma 12.2. *Let k be a positive integer and let N_k be the disequality relation on $[k]$ (so $N_k(x, y)$ if and only if $x \neq y$). The problem* MAX CSP($\{N_k\}$) *is hard to approximate for every $k \geq 2$.*

Proof. MAX CSP($\{N_k\}$) is precisely the MAX k-CUT problem, which is known to be **APX**-complete [9, Problem GT33] and hence it is hard to approximate. □

There is another characterisation of the algebras in Theorem 11.8 which corresponds to tractable constraint languages which we will need in this chapter. To state the characterisation we need the following definition.

Definition 12.3 (Weak Near-Unanimity Function)**.** *An operation $f : D^n \to D$, where $n \geq 2$, is a* weak near-unanimity function *if f is idempotent and*

$$f(x, y, y, \ldots, y) = f(y, x, y, y, \ldots, y) = \ldots = f(y, \ldots, y, x)$$

for all $x, y \in D$.

Hereafter we will use the acronym *wnuf* for weak near-unanimity functions. We say that an algebra $\mathcal{A}$ *admits a wnuf* if there is a wnuf among the term operations of $\mathcal{A}$. We also say that a constraint language Γ admits a wnuf if there is a wnuf among the polymorphisms of Γ. By combining a theorem by Maróti and McKenzie [115, Theorem 1.1] with a result by Bulatov and Jeavons [24, Proposition 4.14], we get the following:

Theorem 12.4. *Let $\mathcal{A}$ be an idempotent algebra. The following are equivalent:*

- *There is a non-trivial factor $\mathcal{B}$ of $\mathcal{A}$ such that $\mathcal{B}$ only has projections as term operations.*
- *The algebra $\mathcal{A}$ does not admit any wnuf.*

12.3 Proof Techniques

In this chapter we will focus on proving that problems are hard to approximate (in the sense of Definition 11.1). To do this we will use AP-reductions and L-reductions. The use of AP-reductions is justified by Lemma 12.5 below. As the existence of an L-reduction implies the existence of an AP-reduction if the problems are in **APX** (this is Lemma 2.5) we are free to use L-reductions when we want to as the problems we are working with here are contained in **APX** (Lemma 12.1).

Lemma 12.5. *If $\Pi_1 \leq_{AP} \Pi_2$ and Π_1 is hard to approximate, then Π_2 is hard to approximate.*

Proof. Let $c > 1$ be the constant such that it is **NP**-hard to approximate Π_1 within c. Let (F, G, α) be the AP-reduction which reduces Π_1 to Π_2. We will prove that it is **NP**-hard to approximate Π_2 within

$$r = \frac{1}{\alpha}(c-1) + 1 - \epsilon'$$

for any $\epsilon' > 0$ such that $r > 1$.

Let I_1 be an instance of Π_1. Then, $I_2 = F(I_1)$ is an instance of Π_2. Given an r-approximate solution to I_2 we can construct an $(1 + (r-1)\alpha + o(1))$-approximate solution to I_1 using G. Hence, we get an $1 + (r-1)\alpha + o(1) = c - \alpha\epsilon' + o(1)$ approximate solution to I_1, and when the instances are large enough this is strictly smaller than c. □

The basic reduction technique in our approximation hardness proofs is based on *strict implementations*. This technique have been used before when studying Max CSP and other CSP-related problems [44, 93, 102].

Definition 12.6 (Implementation)**.** *A collection of constraints $C_1, \ldots, C_m$ over a tuple of variables $\boldsymbol{x} = (x_1, \ldots, x_p)$ called* primary variables *and $\boldsymbol{y} = (y_1, \ldots, y_q)$ called* auxiliary variables *is an* α-implementation *of the p-ary relation R for a positive integer $\alpha \leq m$ if the following conditions are satisfied:*

1. *for any assignment to $\boldsymbol{x}$ and $\boldsymbol{y}$, at most α constraints from $C_1, \ldots, C_m$ are satisfied; and*
2. *for any $\boldsymbol{x}$ such that $\boldsymbol{x} \in R$, there exists an assignment to $\boldsymbol{y}$ such that exactly α constraints are satisfied; and*
3. *for any $\boldsymbol{x}, \boldsymbol{y}$ such that $\boldsymbol{x} \notin R$, at most $(\alpha - 1)$ constraints are satisfied.*

Definition 12.7 (Strict implementation)**.** *An α-implementation is a* strict implementation *if for every $\boldsymbol{x}$ such that $\boldsymbol{x} \notin R$ there exists $\boldsymbol{y}$ such that exactly $(\alpha - 1)$ constraints are satisfied.*

Note that pp-definitions (see Section 2.2) and α-implementations with $\alpha = m$ from Definition 12.7 are the same concept. (In some papers such implementations are called "perfect implementations".)

It will sometimes be convenient for us to view relations as predicates instead. In this case an n-ary relation R over the domain D is a function $r : D^n \to \{0, 1\}$ such that $r(\boldsymbol{x}) = 1 \iff \boldsymbol{x} \in R$. Most of the time we will use predicates when we are dealing with strict implementations and relations when we are working with pp-definitions, because pp-definitions is nothing else than a conjunction of constraints whereas strict implementations may naturally be seen as a sum of predicates. We will write strict α-implementations in the following form

$$g(\boldsymbol{x}) + (\alpha - 1) = \max_{\boldsymbol{y}} \sum_{i=1}^{m} g_i(\boldsymbol{x}_i)$$

where $\boldsymbol{x} = (x_1, \ldots, x_p)$ are the primary variables, $\boldsymbol{y} = (y_1, \ldots, y_q)$ are the auxiliary variables, $g(\boldsymbol{x})$ is the predicate which is implemented, and each $\boldsymbol{x}_i$ is a tuple of variables from $\boldsymbol{x}$ and $\boldsymbol{y}$.

We say that a collection of relations Γ *strictly implements* a relation R if, for some $\alpha \in \mathbb{Z}$, there exists a strict α-implementation of R using relations only from Γ. It is not difficult to show that if R can be obtained from Γ by a series of strict implementations, then it can also be obtained by a single strict implementation (for the boolean case, this is shown in [44, Lemma 5.8]).

The following lemma indicates the importance of strict implementations for MAX CSP. It was first proved for the boolean case, but without the assumption on bounded occurrences, in [44, Lemma 5.17]. A proof of this lemma in our setting can be found in [50, Lemma 3.4] (the lemma is stated in a slightly different form but the proof establishes the required AP-reduction).

Lemma 12.8. *If Γ strictly implements a predicate f, then, for any integer k, there is an integer k' such that* MAX CSP($\Gamma \cup \{f\}$)-$k \leq_{AP}$ MAX CSP(Γ)-k'.

Lemma 12.8 will be used as follows in our proofs of approximation hardness: if Γ' is a fixed finite collection of predicates each of which can be strictly implemented by Γ, then we can assume that $\Gamma' \subseteq \Gamma$. For example, if Γ contains a binary predicate f, then we can assume, at any time when it is convenient, that Γ also contains $f'(x, y) = f(y, x)$, since this equality is a strict 1-implementation of f'.

We will need the following lemma which connects cores and inapproximability.

Lemma 12.9 (Lemma 2.11 in [93]). *Let Γ' be the core of Γ. For every k, there exists k' such that* MAX CSP(Γ')-$k \leq_{AP}$ MAX CSP(Γ)-k'.

The lemma is stated in a slightly different form in [93] but the proof establishes the required AP-reduction. Compared to Lemma 11.4, this lemma connects cores with inapproximability in another way.

12.4 Single Relation MAX CSP

Let $R \in R_D$ be a binary relation. As R is binary it can be viewed as a digraph $G = (V, E)$ with vertex set $V = D$ and edge set $E = R$. We will mix freely between these two notations.

Let $G = (D, E)$ be a digraph, $R = E$, and let $\mathrm{Aut}(G)$ denote the automorphism group of G. If $\mathrm{Aut}(G)$ is transitive (i.e., contains a single orbit), then we say that G is *vertex-transitive*. If D can be partitioned into two sets, A and B, such that for any $x, y \in A$ (or $x, y \in B$) we have $(x, y) \notin R$, then R (and G) is *bipartite*. The *directed cycle of length n* is the digraph G with vertex set $V = \{0, 1, \ldots, n-1\}$ and edge set $\{(x, x+1) \mid x \in V\}$, where the addition is modulo n. Analogously, the *undirected cycle of length n* is the graph H with vertex set V and edge set $\{(x, x+1) \mid x \in V\} \cup \{(x+1, x) \mid x \in V\}$ (also in this case the additions are modulo n). The undirected path with two vertices will be denoted by P_2.

In this section, we will prove the main result of this chapter which is the following theorem:

Theorem 12.10. *Let $R \in R_D^{(n)}$ be non-empty. If $(d, \ldots, d) \in R$ for some $d \in D$, then* MAX CSP($\{R\}$) *is solvable in linear time. Otherwise,* MAX CSP($\{R\}$)-B *is hard to approximate.*

Proof. The tractability part of the theorem is trivial. It was shown in [93] that any non-empty non-valid relation of arity $n \geq 2$ strictly implements a binary non-empty non-valid relation. Hence, by Lemma 12.8, it is sufficient to prove the hardness part for binary relations. We now view the relation as a digraph. The proof for vertex-transitive digraphs is presented in Section 12.4.1, and for the remaining digraphs in Section 12.4.2. □

12.4.1 Vertex-transitive Digraphs

We will now tackle non-bipartite vertex-transitive digraphs and prove that they give rise to MAX CSP problems which are hard at gap location 1. To do this, we make use of the algebraic framework which we used and developed in Section 11.5.

We will also use a theorem by Barto, Kozik, and Niven [10] on the complexity of CSP(G) for digraphs G without sources and sinks. A vertex v in a digraph is a *source* if there is no incoming edge to v. Similarly, a vertex v is a *sink* if there is no outgoing edge from v.

Theorem 12.11 ([10])**.** *If G is a core digraph without sources and sinks which does not retract to a disjoint union of directed cycles, then G admits no wnuf.*

From this result we derive the following corollary.

Corollary 12.12. *Let H be a vertex-transitive core digraph which is non-empty, non-valid, and not a directed cycle. Then,* MAX CSP($\{H\}$)*-B has a hard gap at location 1.*

Proof. Let v and u be two vertices in H. As H is vertex-transitive the in- and out-degrees of u and v must coincide, and hence the in- and out-degrees of v must be the same. Hence, H does not have any sources or sinks. Furthermore, as H is vertex-transitive and a core it follows that it is connected. The result now follows from Theorem 12.11, Theorem 12.4, and Theorem 11.9. □

The next two lemmas will help us deal with the remaining vertex-transitive graphs, i.e., those that retract to a directed cycle.

Lemma 12.13. *If G is an undirected cycle of length $k \geq 2$, then* MAX CSP($\{G\}$)*-B is hard to approximate.*

Proof. If k is even, then the core of G is isomorphic to P_2 and the result follows from Lemmas 12.9, 12.5 combined with Lemma 12.2.

From now on, assume that k is odd, and $k \geq 3$. We will show that we can strictly implement the inequality relation N on the domain $\{0, 1, \ldots, k-1\}$. We use the following strict implementation

$$N(z_1, z_{k-1}) + (k-3) = \max_{z_2, z_3, \ldots, z_{k-2}} G(z_1, z_2) + G(z_2, z_3) + \ldots + G(z_{k-3}, z_{k-2}) + G(z_{k-2}, z_{k-1}).$$

It is not hard to see that if $z_1 \neq z_{k-1}$, then all $k-2$ constraints on the right hand side can be satisfied. If $z_1 = z_{k-1}$, then $k-3$ constraints are satisfied by the assignment $z_i = z_1 + i - 1$, for all i such that $1 < i < k-1$ (the addition and subtraction are modulo k). Furthermore, no assignment can satisfy all constraints. To see this, note that such an assignment would define a path $z_1, z_2, \ldots, z_{k-1}$ in G with $k-2$ edges and $z_1 = z_{k-1}$. This is impossible since $k-2$ is odd and $k-2 < k$.

The lemma now follows from Lemmas 12.8, 12.5, and 12.2. □

Lemma 12.14. *If $G = (V, E)$ is a digraph such that $(x, y) \in E \Rightarrow (y, x) \notin E$, then* MAX CSP($\{H\}$)*-$B \leq_{AP$* MAX CSP($\{G\}$)*-$B$, where H is the undirected graph obtained from G by replacing every edge in G by two edges in opposing directions in H.*

Proof. $H(x, y) + (1-1) = G(x, y) + G(y, x)$ is a strict implementation of H and the result follows from Lemma 12.8. □

Lemma 12.15. *If G is a non-empty non-valid vertex-transitive digraph, then* MAX CSP($\{G\}$)-B *is hard to approximate.*

Proof. By Lemmas 12.9 and 12.5, it is enough to consider cores. For directed cycles, the result follows from Lemmas 12.13 and 12.14, and, for all other digraphs, from Corollary 12.12. □

12.4.2 General Digraphs

We now deal with digraphs that are not vertex-transitive.

Lemma 12.16. *If $G = (V, E)$ is a bipartite digraph which is neither empty nor valid, then* MAX CSP($\{G\}$)-B *is hard to approximate.*

Proof. If there are two edges $(x, y), (y, x) \in E$, then the core of G is isomorphic to P_2 and the result follows from Lemmas 12.5 and 12.9 together with Lemma 12.2. If no such pair of edges exist, then Lemmas 12.5 and 12.14 reduce this case to the previous case where there are two edges $(x, y), (y, x) \in E$. □

We will use a technique known as *domain restriction* [50] in the sequel. The following lemma was proved in [50, Lemma 3.5] (the lemma is stated in a slightly different form there, but the proof together with [9, Lemma 8.2] and Lemma 12.1 implies the existence of the required AP-reduction).

Lemma 12.17. *If $D' \subseteq D$ and $D' \in \Gamma$, then* MAX CSP($\Gamma|_{D'}$)-$B \leq_{AP}$ MAX CSP(Γ)-B.

We are now ready to present the three lemmas that are the building blocks of the main lemma in this section, Lemma 12.22. Let $G = (V, E)$ be a digraph. For a set $X \subseteq V$, we define $X^+ = \{j \mid (i, j) \in E, i \in X\}$, and $X^- = \{i \mid (i, j) \in E, j \in X\}$. When we use Lemma 12.17 we will typically have $D' = \Omega^+$ where Ω is some orbit of $\mathrm{Aut}(G)$. This is justified by Lemma 12.19 and Lemma 12.21 below.

Lemma 12.18. *If a constraint language Γ contains two unary predicates S, T such that $S \cap T = \emptyset$, then Γ strictly implements $S \cup T$.*

Proof. Let $U = S \cup T$. Then $U(x) + (1 - 1) = S(x) + T(x)$ is a strict implementation of $U(x)$. □

Lemma 12.19. *If $G = (D, E)$ is a digraph and there is a pp-definition of $X \subseteq V, X \neq \emptyset$ using G, then G strictly implements X^+ and X^-.*

Proof. Assume that $D = \{1, 2, \ldots, p\}$. Let

$$X(x_1) \iff \exists x_2, \ldots, x_m, y_1, \ldots, y_m : \bigwedge_{i=1}^{m} G(x_i, y_i) \tag{12.1}$$

be a pp-definition of X (here some of the x_i:s and y_i:s may refer to the same variable).

We construct a strict implementation of X^+; the other case can be proved in a similar way. We claim that X^+ can be strictly implemented as follows:

$$X^+(x) + (m+1-1) = \max_{\boldsymbol{x},\boldsymbol{y}} \left(G(x_1, x) + \sum_{i=1}^{m} G(x_i, y_i) \right) \tag{12.2}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$.

Assume first that $x \in X^+$. Choose $x_1 \in X$ such that $G(x_1, x) = 1$. As (12.1) is a pp-definition of X and $x_1 \in X$, it follows that we can choose $x_2, x_3, \ldots, x_m$ and $\boldsymbol{y}$ so that the RHS of (12.2) is $m+1$.

Now consider the case when $x \notin X^+$. Note that if the RHS of (12.2) is $m+1$, then, as (12.1) is a pp-definition of X, it follows that $x_1 \in X$ and hence $x \in X^+$, which is a contradiction. Hence, we can conclude that the RHS of (12.2) is $< m+1$. However, even though $x \notin X^+$, because $X \neq \emptyset$, we can choose $\boldsymbol{x}$ and $\boldsymbol{y}$ so that the RHS of (12.2) is m. □

Lemma 12.20. *If $H = (D, E)$ is a digraph and there is a pp-definition of $X \subseteq D$ using H, then, for every k, there is a k' such that* MAX CSP($\{H|_X\}$)-$k \leq_{AP}$ MAX CSP($\{H\}$)-k'.

Proof. Let $D = \{1, 2, \ldots, p\}$. Let $I = (V, C)$ be an arbitrary instance of MAX CSP($\{H|_X\}$)-k and let $V = \{v_1, \ldots, v_n\}$. We construct an instance $I' = (V' \cup V, C' \cup C)$ of MAX CSP($\{H\}$) as follows: for each variable $v_i \in V$ add k copies of the implementation of $X(v_i)$ to C' and any auxiliary variables used in the implementation are added to V'. It is clear that there is an integer k', which is independent of I, such that I' is an instance of MAX CSP($\{H\}$)-k'.

We prove that MAX CSP($\{H|_X\}$)-$k \leq_L$ MAX CSP($\{H\}$)-k'. It follows from Lemma 2.5 that this is sufficient to establish the lemma as both problems are in **APX** (Lemma 12.1).

Let s' be a solution to I'. For an arbitrary variable $v_i \in V$, if $s(v_i) \notin X$, then at least k constraints in the implementations of $X(v_i)$ are not satisfied by s'. Hence, we can create a new solution, s'', which is identical to s' but where $s''(v_i) \in X$ (we may also need to change the value assigned to some of the auxiliary variables in the implementation of $X(v_i)$). As v_i occurs at most k times in I it follows that $m(I', s') \leq m(I', s'')$. Let us summarise this argument: for any solution s' we can construct another solution $P(s')$ such that $P(s')(v_i) \in X$ for all $v_i \in V$ and $m(I', s') \leq m(I', P(s'))$. Note that for any solution s' to I' the solution $P(s')$ can be seen as a solution to both I and I'.

Let l be the number of constraints used in the implementation of X. By a straightforward probabilistic argument we have $\text{OPT}(I) \geq 1/p^2 \cdot |C|$. Using this fact, the argument above, and the inequality $|V| \leq 2|C|$ we can bound

the optimum of I' as follows:

$$\begin{aligned}\text{OPT}(I') &\leq \text{OPT}(I) + kl \cdot |V| \\ &\leq \text{OPT}(I) + 2kl \cdot |C| \\ &\leq \text{OPT}(I) + 2kp^2 l \cdot \text{OPT}(I) \\ &= (1 + 2kp^2 l)\text{OPT}(I).\end{aligned}$$

Since $P(s')$ satisfies all constraints in C', we have

$$\text{OPT}(I) - m(I, P(s')) = \text{OPT}(I') - m(I', P(s')) \leq \text{OPT}(I') - m(I', s').$$

This establishes the required L-reduction and the lemma follows. □

Lemma 12.21. *Let $H = (D, E)$ be a core digraph and let Ω be an orbit in $\text{Aut}(G)$, then Ω can be pp-defined using H.*

Proof. Assume, without loss of generality, that $D = \{1, 2, \ldots, p\}$ and $1 \in \Omega$. We claim that

$$\Omega(x_1) \iff \exists x_2, x_3, \ldots, x_p : \bigwedge_{(i,j)\in E} H(x_i, x_j). \tag{12.3}$$

Note that if the RHS of (12.3) is satisfied, then the function $i \mapsto x_i$ (that is, the function which maps D to D such that i is mapped to the value assigned to x_i) is a homomorphism of H. (To see this note that if $(i, j) \in E$, then $(x_i, x_j) \in E$ so edges are mapped to edges.) As H is a core any homomorphism from H to H is an automorphism. It follows that $x_1 \in \Omega$.

Conversely, if $x_1 \in \Omega$, then, as Ω is an orbit in H and $1 \in \Omega$, there is an automorphism ρ in $\text{Aut}(G)$ such that $\rho(1) = x_1$. Now, as ρ is an automorphism the assignment $x_i = \rho(i)$ satisfies the RHS of (12.3). □

Lemma 12.22. *Let $H = (V, E)$ be a non-empty non-valid digraph with at least two vertices which is not vertex-transitive. Then* MAX CSP$(\{H\})$*-B is hard to approximate.*

Proof. The proof is by induction on the number of vertices, $|V|$. If $|V| = 2$ then the result follows from Lemma 12.16. Now assume that $|V| > 2$ and the lemma holds for all digraphs with a smaller number of vertices. Note that if H is not a core then the core of H has fewer vertices. The core may be vertex-transitive, but in this case the result follows from Lemma 12.15. If the core of H is not vertex-transitive, then result follows from the induction hypothesis. So we can assume that H is a core.

We claim that either (a) MAX CSP$(\{H\})$-B is hard to approximate, or (b) there exists a proper subset X of V such that $|X| \geq 2$, $H|_X$ is non-empty, $H|_X$ is non-valid and for every k there exists a k' such that MAX CSP$(\{H|_X\})$-$k \leq_{AP}$ MAX CSP$(\{H\})$-k'. Since the core of $H|_X$ has fewer vertices than H, the lemma will follow from this claim, the induction hypothesis, and Lemma 12.15.

We first establish the following claim.

Claim. For any orbits $\Omega_1, \Omega_2 \subseteq V$ such that there is $x \in \Omega_1$ and $y \in \Omega_2$ with $(x, y) \in E$, it follows that $\Omega_2 \subseteq \Omega_1^+$.

Proof. Let Ω_1, Ω_2, x and y be as in the statement of the claim. Let z be an arbitrary vertex in Ω_2. Since Ω_2 is an orbit of H, there is an automorphism $\rho \in \mathrm{Aut}(H)$ such that $\rho(y) = z$, so $(\rho(x), z) \in E$. Furthermore, Ω_1 is an orbit of $\mathrm{Aut}(H)$ so $\rho(x) \in \Omega_1$. Since z was chosen arbitrarily, we conclude that $\Omega_2 \subseteq \Omega_1^+$.

As H is non-empty there are orbits Ω_1 and Ω_2 so that $x \in \Omega_1, y \in \Omega_2$ and $(x, y) \in E$. By the claim above it follows that $\Omega_2 \subseteq \Omega_1^+$.

If $H|_{\Omega_1}$ is non-empty, then we get the result from Lemma 12.21 and Lemma 12.20 and the induction hypothesis, since $\Omega_1 \subsetneq V$ (we cannot have $|\Omega_1| = 1$ because then H would contain a loop). Assume that $H|_{\Omega_1}$ is empty. As $H|_{\Omega_1}$ is empty, we get that Ω_1^+ is a proper subset of V. If $H|_{\Omega_1^+}$ is non-empty, then $|\Omega_1^+| > 1$ and the result follows from Lemma 12.21 (we can pp-define Ω_1), Lemma 12.19 (we can strictly implement Ω_1^+), Lemma 12.8 (strictly implemented relations can freely be added to the constraint language) and Lemma 12.17 (we can restrict our domain to a unary relation). Hence, we assume that $H|_{\Omega_1^+}$ is empty.

Note that $\Omega_1^+ \cap \Omega_2^- = \emptyset$ since $\Omega_2 \subseteq \Omega_1^+$ and $H|_{\Omega_1^+}$ is empty. If $\Omega_1^+ \cup \Omega_2^- \neq V$, then we can use the same sequence of lemmas as above namely, Lemmas 12.21, 12.19, 12.8 and 12.17 together with Lemma 12.18 to get the result we are looking for. This works because $H|_{\Omega_1^+ \cup \Omega_2^-}$ is non-empty. Hence, we can assume without loss of generality that $\Omega_1^+ \cup \Omega_2^- = V$, and since $\Omega_1^+ \cap \Omega_2^- = \emptyset$, we have a partition of V into the sets Ω_1^+ and Ω_2^-. Using the same argument as for Ω_1^+, we can assume that $H|_{\Omega_2^-}$ is empty. Therefore, Ω_1^+,Ω_2^- is a partition of V and $H|_{\Omega_1^+}$ and $H|_{\Omega_2^-}$ are both empty. This implies that H is bipartite and we get the result from Lemma 12.16. □

We will now give a simple corollary to Theorem 12.10 which nevertheless is interesting.

Corollary 12.23. *Let Γ be a constraint language such that $\mathrm{Aut}(\Gamma)$ contains a single orbit. If Γ contains a non-empty k-ary, $k > 1$, relation R which is not d-valid for all $d \in D$, then* MAX CSP(Γ)-B *is hard to approximate. Otherwise,* MAX CSP(Γ) *is tractable.*

Proof. If a relation R with the properties described above exists, then MAX CSP(Γ) is hard to approximate by Theorem 12.10 (note that R cannot be d-valid for any d). Otherwise, every k-ary, $k > 1$, relation $S \in \Gamma$ is d-valid for all $d \in D$. If Γ contains a unary relation U such that $U \subsetneq D$, then $\mathrm{Aut}(\Gamma)$ would contain at least two orbits which contradict our assumptions. It follows that MAX CSP(Γ) is trivially solvable. □

Note that the constraint languages considered in Corollary 12.23 may be seen as a generalisation of vertex-transitive graphs.

12.5 MAX CSP and Non-supermodularity

In this section, we will prove two results whose proofs make use of Theorem 12.10. The first result (Theorem 12.28) concerns the hardness of approximating MAX CSP(Γ) for Γ which contains all at most binary relations which are 2-monotone (see Section 12.5.1 for a definition) on some partially ordered set which is not a lattice order. The other result, Theorem 12.30, states that MAX CSP(Γ) is hard to approximate if Γ contains all at most binary supermodular predicates on some lattice and in addition contains at least one predicate which is not supermodular on the lattice.

These results are interesting as MAX CSP(Γ) has been conjectured to be tractable if and only if the core of Γ is supermodular on some finite lattice (this was discussed a bit in Section 7.5). Hence, with the second result above we know that for any finite lattice $\mathcal{L}$ if the family of all supermodular predicates is tractable for MAX CSP(·), then it is a maximal tractable constraint language for MAX CSP(·). That is, if one adds any relation to the constraint language, then the problem gets hard to approximate.

These results strengthens earlier published results [107, 108] in various ways (e.g., they apply to a larger class of constraint languages or they give approximation hardness instead of **NP**-hardness). In Section 12.5.1 we give a few preliminaries which are needed in this section while the new results are contained in Section 12.5.2.

12.5.1 Preliminaries

The set of all supermodular predicates on a lattice $\mathcal{L}$ will be denoted by $\mathrm{Spmod}_{\mathcal{L}}$. Recall that a constraint language Γ is said to be supermodular on a lattice $\mathcal{L}$ if every predicate in Γ is supermodular on $\mathcal{L}$. In other words Γ is supermodular if $\Gamma \subseteq \mathrm{Spmod}_{\mathcal{L}}$ for some lattice $\mathcal{L}$. We will sometimes use an alternative way of characterising supermodularity:

Theorem 12.24 ([51])**.** *An n-ary function f is supermodular on a lattice $\mathcal{L}$ if and only if it satisfies the supermodular inequality for all $(a_1, a_2, \ldots, a_n)$, $(b_1, b_2, \ldots, b_n) \in \mathcal{L}^n$ such that*

1. *$a_i = b_i$ with one exception, or*
2. *$a_i = b_i$ with two exceptions, and, for each i, the elements a_i and b_i are comparable in $\mathcal{L}$.*

The following definition first occurred in [35].

Definition 12.25 (Generalised 2-monotone)**.** *Given a poset $\mathcal{P} = (D, \sqsubseteq)$, a predicate f is said to be* generalised 2-monotone on $\mathcal{P}$ *if*

$$f(\boldsymbol{x}) = 1 \iff ((x_{i_1} \sqsubseteq a_{i_1}) \wedge \ldots \wedge (x_{i_s} \sqsubseteq a_{i_s})) \vee ((x_{j_1} \sqsupseteq b_{j_1}) \wedge \ldots \wedge (x_{j_s} \sqsupseteq b_{j_s}))$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $a_{i_1}, \ldots, a_{i_s}, b_{j_1}, \ldots, b_{j_s} \in D$, and either of the two disjuncts may be empty.

For brevity, we will use the word *2-monotone* instead of generalised 2-monotone. It is not hard to verify that 2-monotone predicates on some *lattice* are supermodular on the same lattice. It has been shown that for any lattice $\mathcal{L}$ if Γ consists of all 2-monotone predicates over $\mathcal{L}$, then W-MAX CSP(Γ) is in **PO**. [35] In this section we will show that if the poset $\mathcal{P}$ is *not* a lattice order, then the 2-monotone predicates over $\mathcal{P}$ makes MAX CSP($\cdot$) hard to approximate.

The following theorem follows from [50, Remark 4.7]. The proof in [50] uses the corresponding unbounded occurrence case as an essential stepping stone; see [44] for a proof of this latter result.

Theorem 12.26 (MAX CSP on a boolean domain)**.** *Let $D = \{0, 1\}$ and $\Gamma \subseteq R_D$ be a core. If Γ is not supermodular on any lattice on D, then* MAX CSP(Γ)*-B is hard to approximate. Otherwise,* MAX CSP(Γ) *is tractable.*

12.5.2 Results

The following theorem is a combination of results proved in [35] and [107].

Theorem 12.27.

- *If Γ consists of 2-monotone relations on a lattice, then* MAX CSP(Γ) *can be solved in polynomial time.*
- *Let $\mathcal{P} = (D, \sqsubseteq)$ be a poset which is not a lattice. If Γ contains all at most binary 2-monotone relations on $\mathcal{P}$, then* MAX CSP(Γ) *is* **NP***-hard.*

We strengthen the second part of the above result as follows:

Theorem 12.28. *Let $\mathcal{P} = (D, \sqsubseteq)$ be a partial order, which is not a lattice order, on D. If Γ contains all at most binary 2-monotone relations on $\mathcal{P}$, then* MAX CSP(Γ)*-B is hard to approximate.*

Proof. Since $\mathcal{P}$ is a non-lattice partial order, there exist two elements $a, b \in D$ such that either $a \sqcap b$ or $a \sqcup b$ does not exist. We will give a proof for the first case; the other case is analogous.

Let $g(x, y) = 1 \iff (x \sqsubseteq a) \wedge (y \sqsubseteq b)$. The predicate g is 2-monotone on $\mathcal{P}$ so $g \in \Gamma$. We have two cases to consider: (a) a and b have no common lower bound, and (b) a and b have at least two greatest common lower bounds. In the first case g is not valid. To see this, note that if there is an element $c \in D$ such that $g(c, c) = 1$, then $c \sqsubseteq a$ and $c \sqsubseteq b$, and this means that c is a common lower bound for a and b, a contradiction. Hence, g is not valid, and the theorem follows from Theorem 12.10.

In case (b) we will use the domain restriction technique from Lemma 12.17 together with Theorem 12.10. In case (b), there exist two distinct elements $c, d \in D$, such that $c, d \sqsubseteq a$ and $c, d \sqsubseteq b$. Furthermore, we can assume that there is no element $z \in D$ distinct from a, b, c such that $c \sqsubseteq z \sqsubseteq a, b$, and,

similarly, we can assume there is no element $z' \in D$ distinct from a, b, d such that $d \sqsubseteq z' \sqsubseteq a, b$.

Let $f(x) = 1 \iff (x \sqsupseteq c) \wedge (x \sqsupseteq d)$. This predicate is 2-monotone on $\mathcal{P}$. Note that there is no element $z \in D$ such that $f(z) = 1$ and $g(z, z) = 1$, but we have $f(a) = f(b) = g(a, b) = 1$. By restricting the domain to $D' = \{x \in D \mid f(x) = 1\}$ with Lemma 12.17, the result follows from Theorem 12.10. □

The following result was proved in [108].

Theorem 12.29. *Let* Γ *contain all at most binary 2-monotone predicates on some diamond* $\mathcal{M}$*. If* $\Gamma \not\subseteq \mathrm{Spmod}_{\mathcal{M}}$*, then* MAX CSP($\Gamma$) *is* **NP***-hard.*

By modifying the original proof of Theorem 12.29, we can strengthen the result in three ways: our result applies to arbitrary lattices, we prove inapproximability results instead of **NP**-hardness, and we prove the result for bounded occurrence instances.

Theorem 12.30. *Let* Γ *contain all at most binary 2-monotone predicates on an arbitrary lattice* $\mathcal{L}$*. If* $\Gamma \not\subseteq \mathrm{Spmod}_{\mathcal{L}}$*, then* MAX CSP($\Gamma$)-$B$ *is hard to approximate.*

Proof. Let $f \in \Gamma$ be a predicate such that $f \notin \mathrm{Spmod}_{\mathcal{L}}$. We will first prove that f can be assumed to be at most binary. By Theorem 12.24, there is a unary or binary predicate $f' \notin \mathrm{Spmod}_{\mathcal{L}}$ which can be obtained from f by substituting all but at most two variables by constants. We present the initial part of the proof with the assumption that f' is binary, the case when f' is unary can be dealt with in the same way. Denote the constants by $a_3, a_4, \ldots, a_n$ and assume that $f'(x, y) = f(x, y, a_3, a_4, \ldots, a_n)$.

Let $k \geq 5$ be an integer and assume that MAX CSP($\Gamma \cup \{f'\}$)-k is hard to approximate. We will prove that MAX CSP(Γ)-k is hard to approximate by exhibiting an L-reduction from MAX CSP($\Gamma \cup \{f'\}$)-k to MAX CSP(Γ)-k. Given an instance $I = (V, C)$ of MAX CSP($\Gamma \cup \{f'\}$)-k, where $C = \{C_1, C_2, \ldots, C_q\}$, we construct an instance $I' = (V', C')$ of MAX CSP(Γ)-k as follows:

1. for any constraint $(f', \boldsymbol{v}) = C_j \in C$, introduce the constraint $(f, \boldsymbol{v'})$ into C', where $\boldsymbol{v'} = (v_1, v_2, y_3^j, \ldots, y_n^j)$, and add the fresh variables $y_3^j, y_4^j, \ldots, y_n^j$ to V'. Add two copies of the constraints $y_i^j \sqsubseteq a_i$ and $a_i \sqsubseteq y_i^j$ for each $i \in \{3, 4, \ldots, n\}$ to C'.

2. for other constraints, i.e., $(g, \boldsymbol{v}) \in C$ where $g \neq f'$, add $(g, \boldsymbol{v})$ to C'.

It is clear that I' is an instance of MAX CSP(Γ)-k. If we are given a solution s' to I', we can construct a new solution s'' to I' by letting $s''(y_i^j) = a_i$ for all i, j and $s''(x) = s'(x)$, otherwise. Denote this transformation by P, so $s'' = P(s')$. It is not hard to see that $m(I', P(s')) \geq m(I', s')$.

By a simple probabilistic argument there is a constant c, which is independent of I, such that $\mathrm{OPT}(I) \geq c|C|$. (This can be proved in the same

way as we proved Lemma 4.10.) Furthermore, due to the construction of I' and the fact that $m(I', P(s')) \geq m(I', s')$, we have

$$\begin{aligned}\text{OPT}(I') &\leq \text{OPT}(I) + 4(n-2)|C| \\ &\leq \text{OPT}(I) + \frac{4(n-2)}{c} \cdot \text{OPT}(I) \\ &\leq \text{OPT}(I) \cdot \left(1 + \frac{4(n-2)}{c}\right).\end{aligned}$$

Let s' be an r-approximate solution to I'. As $m(I', s') \leq m(I', P(s'))$, we get that $P(s')$ also is an r-approximate solution to I'. Furthermore, since $P(s')$ satisfies all constraints introduced in Step 1, we have

$$\text{OPT}(I) - m(I, P(s')|_V) = \text{OPT}(I') - m(I', P(s')) \leq \text{OPT}(I') - m(I', s').$$

We conclude that MAX CSP($\Gamma \cup \{f'\}$)-k L-reduces to MAX CSP(Γ)-k and hence MAX CSP(Γ)-k is hard to approximate if MAX CSP($\Gamma \cup \{f'\}$)-k is hard to approximate.

We will now prove that MAX CSP(Γ)-B is hard to approximate under the assumption that f is at most binary. We say that the pair $(\boldsymbol{a}, \boldsymbol{b})$ *witnesses the non-supermodularity of* f if $f(\boldsymbol{a}) + f(\boldsymbol{b}) \not\leq f(\boldsymbol{a} \sqcap \boldsymbol{b}) + f(\boldsymbol{a} \sqcup \boldsymbol{b})$.
Case 1: f is unary. As f is not supermodular on $\mathcal{L}$, there exist elements $a, b \in \mathcal{L}$ such that (a, b) witnesses the non-supermodularity of f.

Note that a and b cannot be comparable because we would have $\{a \sqcup b, a \sqcap b\} = \{a, b\}$, and so $f(a \sqcup b) + f(a \sqcap b) = f(a) + f(b)$ contradicting the choice of (a, b). We can now assume, without loss of generality, that $f(a) = 1$. Let $z_* = a \sqcap b$ and $z^* = a \sqcup b$. Note that the two predicates $u(x) = 1 \iff x \sqsubseteq z^*$ and $u'(x) = 1 \iff z_* \sqsubseteq x$ are 2-monotone and, hence, contained in Γ. By using Lemma 12.17, it is therefore sufficient to prove approximation hardness for MAX CSP($\Gamma|_{D'}$)-B, where $D' = \{x \in D \mid z_* \sqsubseteq x \sqsubseteq z^*\}$. We now split the proof into two subcases.
Subcase 1a: $f(a) = 1$ and $f(b) = 1$. At least one of $f(z^*) = 0$ and $f(z_*) = 0$ must hold.

Assume that $f(z_*) = 0$, the other case can be handled in a similar way. Let $g(x, y) = 1 \iff [(x \sqsubseteq a) \wedge (y \sqsubseteq b)]$ and note that g is 2-monotone so $g \in \Gamma$.

Let d be an arbitrary element in D' such that $g(d, d) = 1$. From the definition of g we know that $d \sqsubseteq a, b$ so $d \sqsubseteq z_*$ which implies that $d = z_*$. Furthermore, we have $g(a, b) = 1, f(a) = f(b) = 1$, and $f(z_*) = 0$. Let $D'' = \{x \in D' \mid f(x) = 1\}$. By applying Theorem 12.10 to $g|_{D''}$, we see that MAX CSP($\Gamma|_{D''}$)-B is hard to approximate. Now Lemma 12.17 implies the result for MAX CSP($\Gamma|_{D'}$)-B, and hence for MAX CSP(Γ)-B.
Subcase 1b: $f(a) = 1$ and $f(b) = 0$. In this case $f(z^*) = 0$ and $f(z_*) = 0$.

If there is a $d \in D'$ such that $b \sqsubset d \sqsubset z^*$ and $f(d) = 1$, then we get $f(a) = 1$, $f(d) = 1$, $a \sqcup d = z^*$ and $f(z^*) = 0$, so this case can be handled by Subcase 1a. Assume that such an element d does not exist.

x	y	$t_1(x)$	$t_2(y)$	$g(x,y)$				
0	0	a_1	b_2	0	0	0	0	1
0	1	a_1	a_2	1	1	0	1	1
1	0	b_1	b_2	1	0	1	1	1
1	1	b_1	a_2	1	0	0	0	0

Table 12.1: The five different possibilities for g in Theorem 12.30.

Let $u(x) = 1 \iff b \sqsubseteq x$. The predicate u is 2-monotone so $u \in \Gamma$. Let $h(x) = f|_{D'}(x) + u|_{D'}(x)$. By the observation above, this is a strict implementation. By Lemmas 12.8 and 12.5, it is sufficient to prove the result for $\Gamma' = \Gamma|_{D'} \cup \{h\}$. This can be done exactly as Subcase 1a, with h instead of f. □

Case 2: f is binary. We now assume that Case 1 does not apply. By Theorem 12.24, there exist a_1, a_2, b_1, b_2 such that

$$f(a_1, a_2) + f(b_1, b_2) \not\leq f(a_1 \sqcup b_1, a_2 \sqcup b_2) + f(a_1 \sqcap b_1, a_2 \sqcap b_2) \qquad (12.4)$$

where a_1, b_1 are comparable and a_2, b_2 are comparable. Note that we cannot have $a_1 \sqsubseteq b_1$ and $a_2 \sqsubseteq b_2$, because then the right hand side of (12.4) is equal to $f(b_1, b_2) + f(a_1, a_2)$ which is a contradiction. Hence, we can without loss of generality assume that $a_1 \sqsubseteq b_1$ and $b_2 \sqsubseteq a_2$.

As in Case 1, we will use Lemma 12.17 to restrict our domain. In this case, we will consider the subdomain $D' = \{x \in D \mid z_* \sqsubseteq x \sqsubseteq z^*\}$ where $z_* = a_1 \sqcap b_2$ and $z^* = a_2 \sqcup b_1$. As the two predicates $u_{z^*}(x)$ and $u_{z_*}(x)$, defined by $u_{z^*}(x) = 1 \iff x \sqsubseteq z^*$ and $u_{z_*}(x) = 1 \iff z_* \sqsubseteq x$, are 2-monotone predicates and members of Γ, Lemma 12.17 tells us that it is sufficient to prove hardness for MAX CSP(Γ')-B where $\Gamma' = \Gamma|_{D'}$.

We define two functions, $t_i : \{0,1\} \to \{a_i, b_i\}$ for $i = 1, 2$, as follows:

- $t_1(0) = a_1$ and $t_1(1) = b_1$;
- $t_2(0) = b_2$ and $t_2(1) = a_2$.

Hence, $t_i(0)$ is the least element of a_i and b_i and $t_i(1)$ is the greatest element of a_i and b_i.

Our strategy will be to L-reduce a certain boolean MAX CSP problem to MAX CSP(Γ')-B. Define three boolean predicates as follows: $g(x,y) = f(t_1(x), t_2(y))$, $c_0(x) = 1 \iff x = 0$, and $c_1(x) = 1 \iff x = 1$. From (12.4) it follows that the possibilities for g are quite restricted; the different cases are listed in Table 12.1. One can verify that MAX CSP($\{c_0, c_1, g\}$)-B is hard to approximate for each possible choice of g, by using Theorem 12.26.

The following two 2-monotone predicates (on D') will be used in the reduction

$$h_i(x,y) = 1 \iff [(x \sqsubseteq z_*) \wedge (y \sqsubseteq t_i(0))] \vee [(z^* \sqsubseteq x) \wedge (t_i(1) \sqsubseteq y)]$$

for $i = 1, 2$. The predicates h_1, h_2 are 2-monotone, so they belong to Γ'. We will also use the following predicates

- $L_d(x) = 1 \iff x \sqsubseteq d$,
- $G_d(x) = 1 \iff d \sqsubseteq x$, and
- $N_{d,d'}(x) = 1 \iff (x \sqsubseteq d) \vee (d' \sqsubseteq x)$

for arbitrary $d, d' \in D'$. These predicates are 2-monotone.

Let w be an integer such that MAX CSP$(\{g, c_0, c_1\})$-w is hard to approximate; such an integer exists according to Theorem 12.26. Let $I = (V, C)$, where $V = \{x_1, x_2, \ldots, x_n\}$ and $C = \{C_1, \ldots, C_m\}$, be an instance of MAX CSP$(\{g, c_0, c_1\})$-w. We will construct an instance I' of MAX CSP(Γ')-w', where $w' = 4w + 8$, as follows:

1. for every $C_i \in C$ such that $C_i = ((x_j, x_k), g)$, introduce
   (a) two fresh variables y_j^i and y_k^i,
   (b) the constraint $f(y_j^i, y_k^i)$,
   (c) $w + 2$ copies of the constraints $L_{b_1}(y_j^i), G_{a_1}(y_j^i), N_{a_1,b_1}(y_j^i)$,
   (d) $w + 2$ copies of the constraints $L_{a_2}(y_k^i), G_{b_2}(y_k^i), N_{b_2,a_2}(y_k^i)$, and
   (e) $w + 1$ copies of the constraints $h_1(x_j, y_j^i), h_2(x_k, y_k^i)$; and
2. for every $C_i \in C$ such that $C_i = c_0(x_j)$, introduce the constraint $L_{z_*}(x_j)$; and
3. for every $C_i \in C$ such that $C_i = c_1(x_j)$, introduce the constraint $G_{z^*}(x_j)$.

We can assume that there are no constraints of the form $((x_j, x_j), g)$ in C as any such constraint can either be removed or replaced by (x_j, c_0) or (x_j, c_1).

The intuition behind this construction is as follows: due to the bounded occurrence property and the quite large number of copies of the constraints in Steps 1c, 1d and 1e, all of those constraints will be satisfied in "good" solutions. The elements 0 and 1 in the boolean problem corresponds to z_* and z^*, respectively. This may be seen in the constraints introduced in Steps 2 and 3. The constraints introduced in Step 1c essentially force the variables y_j^i to be either a_1 or b_1, and the constraints in Step 1d work in a similar way. The constraints in Step 1e work as bijective mappings from the domains $\{a_1, b_1\}$ and $\{a_2, b_2\}$ to $\{z_*, z^*\}$, respectively. For example, $h_1(x_j, y_j^i)$ will set x_j to z_* if y_j^i is a_1, otherwise if y_j^i is b_1, then x_j will be set to z^*. Finally, the constraint introduced in Step 1b corresponds to $g(x_j, x_k)$ in the original problem.

It is clear that I' is an instance of MAX CSP(Γ')-w'. Let s' be a solution to I'. If, for some $i \in [m]$ and $j \in [n]$ we have $s'(y_j^i) \notin \{a_1, b_1, a_2, b_2\}$, then

at least $w+2$ constraints from Step 1c or Step 1d are not satisfied by s'. Le s'' be an assignment which is identical to s' except that s'' satisfies all constraints from Step 1c and Step 1d. For each variable of the form y_j^i we change in s'' we satisfy at least $w+2$ new constraints from 1c and 1d. We may make at most $w+2$ constraints from Step 1b and Step 1e unsatisfied in the process, but the end result is that $m(I', s'') \geq m(I', s')$. By a similar argument, we can assume that all constraints from Step 1e are satisfied by s'' (here we use the bounded occurrence property of I).

Given such a solution s'' to I', we can construct a solution $s = G(I, s'')$ to I by, for every $x \in V$, letting $s(x) = 0$ if $s''(x) = z_*$ and $s(x) = 1$, otherwise. Let c' be the constant from Lemma 12.1 such that $\text{OPT}(I) \geq |C|/c'$. It follows that

$$\begin{aligned}\text{OPT}(I') &\leq \text{OPT}(I) + (8w+14)|C| \\ &\leq \text{OPT}(I) + \frac{(8w+14)}{c'} \cdot \text{OPT}(I) \\ &\leq \text{OPT}(I) \cdot \left(1 + \frac{(8w+14)}{c'}\right).\end{aligned}$$

Furthermore, note that for any solution s' to I' we have

$$\text{OPT}(I) - m(I, G(I, s'')) = \text{OPT}(I') - m(I', s'') \leq \text{OPT}(I') - m(I', s').$$

This implies that MAX CSP($\{c_0, c_1, g\}$)-w L-reduces to MAX CSP(Γ')-w' and as MAX CSP($\{c_0, c_1, g\}$)-w is hard to approximate it follows that MAX CSP(Γ')-w' is hard to approximate as well. □

Part IV

Future Work

Chapter 13

Future Work

In this thesis we have investigated the complexity of two optimisation problems related to the CSP, Max Sol and Max CSP. There are, of course, lots of open problems. I would like to take this opportunity to discuss some possible approaches to further progress on these topics and mention some of the open problems.

13.1 Fractional Polymorphisms

Both Max Sol and Max CSP are (essentially) special cases of VCSP. For VCSP there is an algebraic characterisation of the expressibility of a valued constraint language, resembling the polymorphisms and the associated algebras one has for constraint languages for CSP. The concept corresponding to polymorphisms in the CSP case is called *fractional polymorphisms* in the VCSP case. It is known that the fractional polymorphisms associated with a valued constraint language decides the computational complexity of the corresponding VCSP [33]. The submodularity condition that we investigated in Chapters 8–10 is in fact a fractional polymorphism. General fractional polymorphisms may be seen as a substantial generalisation of submodular functions.

I believe that studying Max Sol and Max CSP from the VCSP perspective will be fruitful for the progress of the area. In particular, by finding new classes of fractional polymorphisms which imply tractability one obtains new classes of tractable constraint languages.

Another, related approach, is to try to come up with an analogue to the algebraic CSP dichotomy conjecture (the conjecture that Theorem 11.8 captures all **NP**-hard cases for CSP) for the VCSP case. By using fractional polymorphisms it may be possible to show that various constructions preserve tractability. In particular, taking homomorphic images, some appropriate generalisation of Mal'tsev products and (perhaps in some way restricted) subalgebra constructions are candidates for preserving tractabil-

ity for general fractional polymorphisms. This is very much in line with what we did in Chapter 10 for submodular functions.

13.2 Combining Soft and Crisp Constraints

Raghavendra's result [127], which was mentioned in Section 7.1, is very impressive. (Recall that Raghavendra proved almost tight approximability results for VCSP(Γ) for all Γ such that there is no $f \in \Gamma$ and tuple $\boldsymbol{t}$ such that $f(\boldsymbol{t}) = -\infty$.) When one formulates Max Sol as a VCSP one ends up with a valued constraint language with such "crisp" constraints, i.e., there will be cost functions which maps tuples to $-\infty$. Hence, Raghavendra's result does not apply in this case. On the other hand, all the work done on the complexity of CSP, when seen in the VCSP setting, does only apply to valued constraint languages where the cost functions have range $\{0, -\infty\}$.

Is it possible to combine the semidefinite programming approach of [127] with the approaches used for results on CSP to gain an understanding of the complexity of VCSP with a combination of "soft" and "crisp" constraints? In particular, how can the approximability of VCSP be dealt with?

13.3 SFM on Diamonds in Polynomial Time

In Chapter 8 we constructed a pseudopolynomial-time algorithm for the SFM problem on diamonds. It would be desirable to have a polynomial-time algorithm instead, i.e., showing that $\mathcal{M}_k$ is oracle-tractable for every $k \geq 3$. One possible approach may be to use some kind of scaling technique see, e.g., [84, 87].

13.4 Is $P(f)$ $1/k$-integral?

In Section 9.5 we mentioned that one approach of establishing oracle-pseudo-tractability for modular atomistic lattices was to show that for each modular atomistic lattice $\mathcal{L}$ there is some integer $k(\mathcal{L})$ such that if $f : \mathcal{L}^n \to \mathbb{Z}$ is submodular, then $P(f)$ is $1/k(\mathcal{L})$-integral. Recall that by "$1/k(\mathcal{L})$-integral" we mean that if $\boldsymbol{x}$ is a vertex of $P(f)$, then each coordinate of $\boldsymbol{x}$ is contained in $\{1/k(\mathcal{L}) \cdot m \mid m \in \mathbb{Z}\}$.

13.5 Avoiding the Ellipsoid Algorithm

The algorithms constructed in Chapters 8–10 used the Ellipsoid algorithm as a subroutine. Unfortunately, even though the Ellipsoid algorithm runs in polynomial time it is considered to be too slow to be practically useful. (The algorithm for diamonds in Chapter 8 actually consists of a *nested* application of the Ellipsoid algorithm. Usually, one layer of the Ellipsoid

algorithm is considered to be too inefficient to be used in practise.) Can one come up with a combinatorial algorithm, i.e., one that does not use the Ellipsoid algorithm, for the problems we considered in Chapters 8–10? For the submodular set function case the first polynomial time algorithm was also based on the Ellipsoid algorithm and then, almost twenty years later, several combinatorial algorithms were developed. No combinatorial algorithms are known for submodular function minimisation over other lattices, except the ones which can be constructed by repeatedly taking homomorphic images, Mal'tsev products and sublattices of distributive lattices, i.e., MPVAR($\mathbf{D}_{\text{fin}}$).

13.6 SFM over Sublattices

Krokhin and Larose [109] asked if it is true for general finite lattices that if $\mathcal{S}$ is a sublattice of $\mathcal{L}$ and SFM($\mathcal{L}$) is oracle-tractable, does it follow that SFM($\mathcal{S}$) is oracle-tractable? In Chapter 10 we showed that this holds for all modular lattices. Somewhat related to this issue is our observation in Chapter 10 that if one starts with a class of lattices which is closed under taking sublattices, then the class obtained by repeatedly taking homomorphic images and Mal'tsev products also is closed under taking sublattices. However, in general it is an open problem if any of the non-hardness notions are closed under taking sublattices.

The open problem in this area is to show that SFM($\mathcal{L}$) is oracle-tractable for all finite lattices $\mathcal{L}$. Such a result would of course immediately give an affirmative answer to Krokhin and Larose's question.

13.7 Approximability Thresholds for Hard CSPs

In Chapter 11 we showed that Max CSP(Γ) has a hard gap at location 1 whenever Γ satisfies a certain condition which makes CSP(Γ) **NP**-hard. However, we paid no attention to the constant which we prove inapproximability for. The following is thus an open problem: given a constraint language Γ, what is the largest constant $c(\Gamma)$ such that it is **NP**-hard to tell completely satisfiable instances from instances were at most a $c(\Gamma)$-fraction of the constraints are satisfiable for Max CSP(Γ)?

For some constraint languages good bounds are known. One example is Max 3SAT for which Johan Håstad showed that it is **NP**-hard to tell satisfiable instances from those where only a $(7/8 + \epsilon)$-fraction of the constraints are satisfiable, for an arbitrary small $\epsilon > 0$. [81] This result is tight as a random assignment satisfies a 7/8-fraction in expectation. Another example is the so-called Not-Two constraint language for which a conditional tight approximability result is know [119]. We note that Raghavendra's result [127], which gives tight approximability results for many VCSPs under the assumption of the UGC, is not applicable in the case when one wants

to tell completely satisfiable instances from instances were only a fraction of the constraints are satisfiable.

Bibliography

[1] P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theor. Comput. Sci.*, 237(1-2):123–134, 2000.

[2] J.-C. Anglès-d'Auriac, F. Iglói, M. Preissmann, and Á. Sebõ. Optimal cooperation and submodularity for computing Potts' partition functions with a large number of states. *Journal of Physics A: Mathematical and General*, 35:6973–6983(11), 2002.

[3] G. Appa and B. Kotnyek. Rational and integral k-regular matrices. *Discrete Mathematics*, 275(1-3):1–15, 2004.

[4] K. R. Apt. *Principles of Constraint Programming*. Cambridge University Press, 2003.

[5] S. Arora. *Probabilistic checking of proofs and hardness of approximation problems*. PhD thesis, University of California at Berkeley, 1995.

[6] S. Arora and B. Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, 2009.

[7] S. Arora and C. Lund. Hardness of approximations. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, chapter 10, pages 399–446. PWS Publishing, Boston, MA, USA, 1997.

[8] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[9] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation: Combinatorial Optimization Problems and their Approximability Properties*. Springer, 1999.

[10] L. Barto, M. Kozik, and T. Niven. The CSP dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of Bang-Jensen and Hell). *SIAM J. Comput.*, 38(5):1782–1802, 2009.

[11] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3289^n)$. *J. Algorithms*, 54(2):168–204, February 2005.

[12] P. Berman and T. Fujito. On the approximation properties of independent set problem in degree 3 graphs. In *WADS '95*, pages 449–460, 1995.

[13] G. Birkhoff. *Lattice theory. 3rd ed.* American Mathematical Society Colloquium Publications, New York, 1967.

[14] M. Bodirsky. Constraint satisfaction problems with infinite templates. In N. Creignou, P. G. Kolaitis, and H. Vollmer, editors, *Complexity of Constraints: An Overview of Current Research Themes*, pages 196–228. Springer-Verlag, Berlin, Heidelberg, 2008.

[15] A. Bouchet. Matchings and Δ-matroids. *Discrete Appl. Math.*, 24(1-3):55–62, 1989.

[16] A. Bouchet and W. H. Cunningham. Delta-matroids, jump systems, and bisubmodular polyhedra. *SIAM J. Discret. Math.*, 8(1):17–32, 1995.

[17] A. Bulatov. Tractable conservative constraint satisfaction problems. Submitted to ACM Trans. Comput. Log.

[18] A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS '02)*, pages 649–658, Washington, DC, USA, 2002. IEEE Computer Society.

[19] A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 321–330, Washington, DC, USA, 2003. IEEE Computer Society.

[20] A. Bulatov. A graph of a relational structure and constraint satisfaction problems. In *LICS '04: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, pages 448–457, Washington, DC, USA, 2004. IEEE Computer Society.

[21] A. Bulatov. Combinatorial problems raised from 2-semilattices. *J. Algebra*, 298(2):321–339, 2006.

[22] A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006.

[23] A. Bulatov and V. Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27, 2006.

[24] A. Bulatov and P. Jeavons. Algebraic structures in combinatorial problems. Technical Report MATH-AL-4-2001, Technische Universität Dresden, 2001.

[25] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005.

[26] A. Bulatov, A. Krokhin, and P. Jeavons. The complexity of maximal constraint languages. In *Proceedings of the thirty-third Annual ACM Symposium on Theory of Computing (STOC '01)*, pages 667–674, New York, NY, USA, 2001. ACM Press.

[27] S. Burris and H. Sankappanavar. *A Course in Universal Algebra*. Springer Verlag, Berlin, 1981.

[28] E. Böhler, S. Reith, H. Schnoor, and H. Vollmer. Bases for boolean co-clones. *Information Processing Letters*, 96(2):59–66, 2005.

[29] C. Carathéodory. Über den variabilitätsbereich der fourierschen konstanten von positiven harmonischen funktionen. *Rend. Circ. Mat. Palermo*, 32:193–217, 1911.

[30] R. D. Carr and O. Parekh. A 1/2-integral relaxation for the A-matching problem. *Oper. Res. Lett.*, 34(4):445–450, 2006.

[31] H. Chen. Quantified constraint satisfaction, maximal constraint languages, and symmetric polymorphisms. In *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS '05)*, pages 315–326. Springer, 2005.

[32] D. Cohen. Tractable decision for a constraint language implies tractable search. *Constraints*, 9(3):219–229, 2004.

[33] D. Cohen, M. Cooper, and P. Jeavons. An algebraic characterisation of complexity for valued constraints. In *Proceedings of the 12th International Conference on Principles and Practice of Contraint Programming (CP'06)*, volume 4204 of *Lecture Notes in Computer Science*, pages 107–121, 2006.

[34] D. Cohen, M. Cooper, and P. Jeavons. Generalising submodularity and horn clauses: tractable optimization problems defined by tournament pair multimorphisms. *Theor. Comput. Sci.*, 401(1-3):36–51, 2008.

[35] D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin. Supermodular functions and the complexity of Max CSP. *Discrete Appl. Math.*, 149(1-3):53–72, 2005.

[36] D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin. The complexity of soft constraint satisfaction. *Artificial Intelligence*, 170(11):909–1030, 2006.

[37] D. Cohen and P. Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 8, pages 245–280. Elsevier, 2006.

[38] P. Cohn. *Universal Algebra*. Number 6 in Mathematics and its Applications. Reidel, 1981. Originally published by Harper and Row, 1965.

[39] M. Cooper. Minimization of locally defined submodular functions by optimal soft arc consistency. *Constraints*, 13(4):437–458, 2008.

[40] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[41] G. Cornuéjols. General factors of graphs. *J. Comb. Theory Ser. B*, 45(2):185–198, 1988.

[42] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *J. Comput. Syst. Sci.*, 51(3):511–522, 1995.

[43] N. Creignou, M. Hermann, A. Krokhin, and G. Salzer. Complexity of clausal constraints over chains. *Theor. Comp. Sys.*, 42(2):239–255, 2008.

[44] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[45] N. Creignou, P. Kolaitis, and B. Zanuttini. Structure identification of boolean relations and plain bases for co-clones. *Journal of Computer and System Sciences*, 74(7):1103–1115, 2008.

[46] P. Crescenzi. A short guide to approximation preserving reductions. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity (CCC '97)*, pages 262–273, Washington, DC, USA, 1997. IEEE Computer Society.

[47] V. Dalmau and D. Ford. Generalized satisfability with limited occurrences per variable: A study through delta-matroid parity. In *MFCS '03*, pages 358–367, 2003.

[48] V. Dalmau and P. Jeavons. Learnability of quantified formulas. *Theor. Comput. Sci.*, 306(1-3):485–511, 2003.

[49] V. Dalmau and J. Pearson. Closure functions and width 1 problems. In *CP '99: Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming*, pages 159–173, London, UK, 1999. Springer-Verlag.

[50] V. Deineko, P. Jonsson, M. Klasson, and A. Krokhin. The approximability of Max CSP with fixed-value constraints. *J. ACM*, 55(4):1–37, 2008.

[51] B. L. Dietrich and A. J. Hoffman. On greedy algorithms, partially ordered sets, and submodular functions. *IBM J. Res. Dev.*, 47(1):25–30, 2003.

[52] I. Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[53] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications*. Gordon and Breach, 1970.

[54] J. Edmonds and E. L. Johnson. Matching: A well-solved class of integer linear programs. In *Combinatorial Optimization – Eureka, You Shrink!*, pages 27–30, 2001.

[55] L. Engebretsen, J. Holmerin, and A. Russell. Inapproximability results for equations over finite groups. *Theor. Comput. Sci.*, 312(1):17–45, 2004.

[56] T. Feder. Fanout limitations on constraint systems. *Theor. Comput. Sci.*, 255(1-2):281–293, 2001.

[57] T. Feder and D. Ford. Classification of bipartite boolean constraint satisfaction through delta-matroid intersection. Technical Report TR05-016, ECCC, 2005.

[58] T. Feder and D. Ford. Classification of bipartite boolean constraint satisfaction through delta-matroid intersection. *SIAM J. Discret. Math.*, 20(2):372–394, 2006.

[59] T. Feder, P. Hell, and J. Huang. List homomorphisms of graphs with bounded degrees. *Discrete Math.*, 307:386–392, 2007.

[60] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1999.

[61] I. Fleischer and T. Traynor. Group-valued modular functions. *Algebra Universalis*, 14:287–291, 1982.

[62] A. Frank. Submodular functions in graph theory. *Discrete Math.*, 111(1-3):231–243, 1993.

[63] S. Fujishige. Polymatroidal dependence structure of a set of random variables. *Information and Control*, 39(1):55–72, 1978.

[64] S. Fujishige. Theory of submodular programs: A Fenchel-type min-max theorem and subgradients of submodular functions. *Math. Program.*, 29(2):142–155, 1984.

[65] S. Fujishige. *Submodular Functions and Optimization*, volume 58 of *Ann. Discrete Math.* Elsevier Science, 2 edition, 2005.

[66] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[67] D. Gijswijt and G. Pap. An algorithm for weighted fractional matroid matching. Technical report, EGRES TR-2008-11, MTA-ELTE Egerváry Research Group on Combinatorial Optimization, Dept. of Operations Research, Eötvös University, Budapest, 2008.

[68] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[69] M. Goldmann and A. Russell. The complexity of solving equations over finite groups. *Inf. Comput.*, 178(1):253–262, 2002.

[70] G. Grätzer and D. Kelly. Products of lattice varieties. *Algebra Universalis*, 21:33–45, 1984.

[71] G. Grätzer and D. Kelly. A survey of products of lattice varieties. *Coll. Math. Soc. J. Bolyai 33*, pages 457–469, 1980.

[72] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[73] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[74] T. S. Han. The capacity region of a general multiple access channel with certain correlated sources. *Inf. Control*, 40:37–60, 1979.

[75] P. Hell and J. Nešetřil. On the complexity of H-coloring. *J. Comb. Theory Ser. B*, 48(1):92–110, 1990.

[76] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[77] D. Hochbaum, N. Megiddo, J. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Math. Program.*, 62:69–83, 1993.

[78] D. Hochbaum and J. Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM J. Comput.*, 23(6):1179–1192, 1994.

[79] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[80] J. Håstad. Every 2-CSP allows nontrivial approximation. In H. N. Gabow and R. Fagin, editors, *STOC*, pages 740–746. ACM, 2005.

[81] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[82] J. Håstad. Personal communication, 2005.

[83] G. Istrate. Looking for a version of Schaefer's dichotomy theorem when each variable occurs at most twice. Technical Report TR652, University of Rochester Computer Science Department, 1997.

[84] S. Iwata. A capacity scaling algorithm for convex cost submodular flows. In *Proceedings of the seventh Annual ACM-SIAM Symposium on Discrete algorithms (SODA '96)*, pages 482–489, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.

[85] S. Iwata. A faster scaling algorithm for minimizing submodular functions. In *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization (IPCO '02)*, volume 2337 of *Lecture Notes in Computer Science*, pages 1–8. Springer-Verlag, 2002.

[86] S. Iwata. Submodular function minimization. *Math. Program.*, 112:45–64, 2008.

[87] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 48(4):761–777, 2001.

[88] P. Janssen, P. Jegou, B. Nouguier, and M. Vilarem. A filtering process for general constraint satisfaction problems: achieving pair-wise consistency using an associated binary representation. In *Proceedings of the IEEE Workshop on Tools for Artificial Intelligence*, pages 420–427, 1989.

[89] P. Jeavons. On the algebraic structure of combinatorial problems. *Theor. Comput. Sci.*, 200(1-2):185–204, 1998.

[90] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *J. ACM*, 44:527–548, 1997.

[91] P. Jeavons and M. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79:327–339, 1996.

[92] P. Jipsen and H. Rose. *Varieties of Lattices*, volume 1533 of *Lecture Notes in Mathematics*. Springer, 1992.

[93] P. Jonsson, M. Klasson, and A. Krokhin. The approximability of three-valued Max CSP. *SIAM J. Comput.*, 35(6):1329–1349, 2006.

[94] P. Jonsson and A. Krokhin. Maximum H-colourable subdigraphs and constraint optimization with arbitrary weights. *J. Comput. System Sci.*, 73(5):691–702, 2007.

[95] P. Jonsson, A. Krokhin, and F. Kuivinen. Ruling out polynomial-time approximation schemes for hard constraint satisfaction problems. In *Proceedings of the Second International Symposium on Computer Science in Russia (CSR '07)*, pages 182–193. Springer, 2007.

[96] P. Jonsson, F. Kuivinen, and G. Nordh. Approximability of integer programming with generalised constraints. In *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP '06)*, pages 256–270. Springer, 2006.

[97] P. Jonsson, G. Nordh, and J. Thapper. The maximum solution problem on graphs. In L. Kucera and A. Kucera, editors, *MFCS*, volume 4708 of *Lecture Notes in Computer Science*, pages 228–239. Springer, 2007.

[98] T. W. Judson. *Abstract Algebra, Theory and Applications*. PWS Publishing Company, 1994.

[99] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Inf. Process. Lett.*, 37(1):27–35, 1991.

[100] R. Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–104. Plenum Press, 1972.

[101] S. Khanna, R. Motwani, M. Sudan, and U. V. Vazirani. On syntactic versus computational views of approximability. *SIAM J. Comput.*, 28(1):164–191, 1998.

[102] S. Khanna, M. Sudan, L. Trevisan, and D. P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.

[103] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth Annual ACM Symposium on Theory of Computing (STOC '02)*, pages 767–775, New York, NY, USA, 2002. ACM Press.

[104] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for Max-Cut and other 2-variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007.

[105] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *Journal of Computer and System Sciences*, 74(3):335 – 349, 2008.

[106] J. Kratochvíl, P. Savický, and Z. Tuza. One more occurrence of variables makes satisfiability jump from trivial to NP-complete. *SIAM J. Comput.*, 22(1):203–210, 1993.

[107] A. Krokhin and B. Larose. Maximum constraint satisfaction on diamonds. Technical Report CS-RR-408, University of Warwick, UK, 2004.

[108] A. Krokhin and B. Larose. Maximum constraint satisfaction on diamonds. In *Principles and Practice of Constraint Programming (CP '05)*, pages 388–402. Springer, 2005.

[109] A. Krokhin and B. Larose. Maximizing supermodular functions on product lattices, with application to maximum constraint satisfaction. *SIAM J. Discret. Math.*, 22(1):312–328, 2008.

[110] R. E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

[111] B. Larose and L. Zádori. Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras. *Internat. J. Algebra Comput.*, 16(3):563–581, 2006.

[112] L. Lovász. Submodular functions and convexity. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming—The State of the Art*, pages 235–257. Springer, 1982.

[113] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[114] A. I. Mal'tsev. Multiplication of classes of algebraic systems (russian). *Siberian Math. J.*, 8:254–267, 1967. English translation: *The Metamathematics of Algebraic Systems*, North-Holland, Amsterdam, 1971, 422–446.

[115] M. Maróti and R. McKenzie. Existence theorems for weakly symmetric operations. *Algebra Universalis*, 59(3–4):463–489, 2008.

[116] S. T. McCormick. Submodular function minimization, 2007. Version 3a. Based on Chapter 7 of the *Handbook on Discrete Optimization*, Elsevier, K. Aardal, G. Nemhauser, and R. Weismantel, editors, 321–391.

[117] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[118] G. Nordh and P. Jonsson. Approximability of clausal constraints. *Theory of computing systems*, 2008. To appear.

[119] R. O'Donnell and Y. Wu. Conditional hardness for satisfiable 3-CSPs. In *Proceedings of the 41st annual ACM symposium on Theory of computing (STOC '09)*, pages 493–502, New York, NY, USA, 2009. ACM.

[120] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. In *Integer Programming and Combinatorial Optimization, 12th International IPCO Conference (IPCO '07)*, volume 4513 of *Lecture Notes in Computer Science*, pages 240–251. Springer-Verlag, 2007.

[121] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[122] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the twentieth Annual ACM Symposium on Theory of Computing (STOC '88)*, pages 229–234, New York, NY, USA, 1988. ACM Press.

[123] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. System Sci.*, 43:425–440, 1991.

[124] E. Petrank. The hardness of approximation: Gap location. *Computational Complexity*, 4:133–157, 1994.

[125] E. Post. The two-valued iterative systems of mathematical logic. *Ann. of Math. Stud.*, 5:1–122, 1941.

[126] R. Pöschel and L. Kalužnin. *Funktionen- und Relationenalgebren*. DVW, Berlin, 1979.

[127] P. Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*, pages 245–254, New York, NY, USA, 2008. ACM Press.

[128] I. Rosenberg. Minimal clones I: the five types. In L. Szabó and Á. Szendrei, editors, *Lectures in Universal Algebra*, pages 405–427. North-Holland, 1986.

[129] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[130] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth Annual ACM Symposium on Theory of Computing (STOC '78)*, pages 216–226, New York, NY, USA, 1978. ACM Press.

[131] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI '95)*, volume 1, pages 631–639, 1995.

[132] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[133] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory Ser. B*, 80(2):346–355, 2000.

[134] A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer, 2003.

[135] L. S. Shapley. Cores of convex games. *Internat. J. Game Theory*, 1(1):11–26, 1971.

[136] M. Sipser. *Introduction to the Theory of Computation, Second Edition*. Course Technology, 2005.

[137] B. Szczepara. *Minimal clones generated by groupoids*. PhD thesis, Université de Móntreal, 1996.

[138] Á. Szendrei. *Clones in Universal Algebra*, volume 99 of *Séminaire de Mathématiques Supérieures*. University of Montreal, 1986.

[139] Á. Szendrei. Idempotent algebras with restrictions on subalgebras. *Acta Sci. Math. (Szeged)*, 51(1–2):57–65, 1987.

[140] L. Trevisan. Inapproximability of combinatorial optimization problems, 2004. arXiv.org:cs.CC/0409043.

[141] G. Woeginger. An efficient algorithm for a class of constraint satisfaction problems. *Oper. Res. Lett.*, 30(1):9–16, 2002.

[142] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.

Department of Computer and Information Science
Linköpings universitet

Dissertations

Linköping Studies in Science and Technology

No 14 **Anders Haraldsson:** A Program Manipulation System Based on Partial Evaluation, 1977, ISBN 91-7372-144-1.

No 17 **Bengt Magnhagen:** Probability Based Verification of Time Margins in Digital Designs, 1977, ISBN 91-7372-157-3.

No 18 **Mats Cedwall**: Semantisk analys av process-beskrivningar i naturligt språk, 1977, ISBN 91- 7372-168-9.

No 22 **Jaak Urmi:** A Machine Independent LISP Compiler and its Implications for Ideal Hardware, 1978, ISBN 91-7372-188-3.

No 33 **Tore Risch:** Compilation of Multiple File Queries in a Meta-Database System 1978, ISBN 91- 7372-232-4.

No 51 **Erland Jungert:** Synthesizing Database Structures from a User Oriented Data Model, 1980, ISBN 91-7372-387-8.

No 54 **Sture Hägglund:** Contributions to the Development of Methods and Tools for Interactive Design of Applications Software, 1980, ISBN 91-7372-404-1.

No 55 **Pär Emanuelson:** Performance Enhancement in a Well-Structured Pattern Matcher through Partial Evaluation, 1980, ISBN 91-7372-403-3.

No 58 **Bengt Johnsson, Bertil Andersson:** The Human-Computer Interface in Commercial Systems, 1981, ISBN 91-7372-414-9.

No 69 **H. Jan Komorowski:** A Specification of an Abstract Prolog Machine and its Application to Partial Evaluation, 1981, ISBN 91-7372-479-3.

No 71 **René Reboh:** Knowledge Engineering Techniques and Tools for Expert Systems, 1981, ISBN 91-7372-489-0.

No 77 **Östen Oskarsson:** Mechanisms of Modifiability in large Software Systems, 1982, ISBN 91- 7372-527-7.

No 94 **Hans Lunell:** Code Generator Writing Systems, 1983, ISBN 91-7372-652-4.

No 97 **Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983, ISBN 91-7372-660-5.

No 109 **Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation, 1984, ISBN 91-7372-801-2.

No 111 **Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984, ISBN 91-7372- 805-5.

No 155 **Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons, 1987, ISBN 91-7870-133-3.

No 165 **James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning, 1987, ISBN 91-7870-183-X.

No 170 **Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems, 1987, ISBN 91-7870-225-9.

No 174 **Johan Fagerström:** A Paradigm and System for Design of Distributed Systems, 1988, ISBN 91-7870-301-8.

No 192 **Dimiter Driankov:** Towards a Many Valued Logic of Quantified Belief, 1988, ISBN 91-7870-374-3.

No 213 **Lin Padgham:** Non-Monotonic Inheritance for an Object Oriented Knowledge Base, 1989, ISBN 91-7870-485-5.

No 214 **Tony Larsson:** A Formal Hardware Description and Verification Method, 1989, ISBN 91-7870-517-7.

No 221 **Michael Reinfrank:** Fundamentals and Logical Foundations of Truth Maintenance, 1989, ISBN 91-7870-546-0.

No 239 **Jonas Löwgren:** Knowledge-Based Design Support and Discourse Management in User Interface Management Systems, 1991, ISBN 91-7870-720-X.

No 244 **Henrik Eriksson:** Meta-Tool Support for Knowledge Acquisition, 1991, ISBN 91-7870-746-3.

No 252 **Peter Eklund:** An Epistemic Approach to Interactive Design in Multiple Inheritance Hierarchies, 1991, ISBN 91-7870-784-6.

No 258 **Patrick Doherty:** NML3 - A Non-Monotonic Formalism with Explicit Defaults, 1991, ISBN 91-7870-816-8.

No 260 **Nahid Shahmehri:** Generalized Algorithmic Debugging, 1991, ISBN 91-7870-828-1.

No 264 **Nils Dahlbäck:** Representation of Discourse-Cognitive and Computational Aspects, 1992, ISBN 91-7870-850-8.

No 265 **Ulf Nilsson:** Abstract Interpretations and Abstract Machines: Contributions to a Methodology for the Implementation of Logic Programs, 1992, ISBN 91-7870-858-3.

No 270 **Ralph Rönnquist:** Theory and Practice of Tense-bound Object References, 1992, ISBN 91-7870-873-7.

No 273 **Björn Fjellborg:** Pipeline Extraction for VLSI Data Path Synthesis, 1992, ISBN 91-7870-880-X.

No 276 **Staffan Bonnier:** A Formal Basis for Horn Clause Logic with External Polymorphic Functions, 1992, ISBN 91-7870-896-6.

No 277 **Kristian Sandahl:** Developing Knowledge Management Systems with an Active Expert Methodology, 1992, ISBN 91-7870-897-4.

No 281 **Christer Bäckström:** Computational Complexity of Reasoning about Plans, 1992, ISBN 91-7870-979-2.

No 292 **Mats Wirén:** Studies in Incremental Natural Language Analysis, 1992, ISBN 91-7871-027-8.

No 297 **Mariam Kamkar:** Interprocedural Dynamic Slicing with Applications to Debugging and Testing, 1993, ISBN 91-7871-065-0.

No 302 **Tingting Zhang:** A Study in Diagnosis Using Classification and Defaults, 1993, ISBN 91-7871-078-2

No 312 **Arne Jönsson:** Dialogue Management for Natural Language Interfaces - An Empirical Approach, 1993, ISBN 91-7871-110-X.

No 338 **Simin Nadjm-Tehrani:** Reactive Systems in Physical Environments: Compositional Modelling and Framework for Verification, 1994, ISBN 91-7871-237-8.

No 371 **Bengt Savén:** Business Models for Decision Support and Learning. A Study of Discrete-Event Manufacturing Simulation at Asea/ABB 1968-1993, 1995, ISBN 91-7871-494-X.

No 375 **Ulf Söderman:** Conceptual Modelling of Mode Switching Physical Systems, 1995, ISBN 91-7871-516-4.

No 383 **Andreas Kågedal:** Exploiting Groundness in Logic Programs, 1995, ISBN 91-7871-538-5.

No 396 **George Fodor:** Ontological Control, Description, Identification and Recovery from Problematic Control Situations, 1995, ISBN 91-7871-603-9.

No 413 **Mikael Pettersson:** Compiling Natural Semantics, 1995, ISBN 91-7871-641-1.

No 414 **Xinli Gu:** RT Level Testability Improvement by Testability Analysis and Transformations, 1996, ISBN 91-7871-654-3.

No 416 **Hua Shu:** Distributed Default Reasoning, 1996, ISBN 91-7871-665-9.

No 429 **Jaime Villegas:** Simulation Supported Industrial Training from an Organisational Learning Perspective - Development and Evaluation of the SSIT Method, 1996, ISBN 91-7871-700-0.

No 431 **Peter Jonsson:** Studies in Action Planning: Algorithms and Complexity, 1996, ISBN 91-7871-704-3.

No 437 **Johan Boye:** Directional Types in Logic Programming, 1996, ISBN 91-7871-725-6.

No 439 **Cecilia Sjöberg:** Activities, Voices and Arenas: Participatory Design in Practice, 1996, ISBN 91-7871-728-0.

No 448 **Patrick Lambrix:** Part-Whole Reasoning in Description Logics, 1996, ISBN 91-7871-820-1.

No 452 **Kjell Orsborn:** On Extensible and Object-Relational Database Technology for Finite Element Analysis Applications, 1996, ISBN 91-7871-827-9.

No 459 **Olof Johansson:** Development Environments for Complex Product Models, 1996, ISBN 91-7871-855-4.

No 461 **Lena Strömbäck:** User-Defined Constructions in Unification-Based Formalisms, 1997, ISBN 91-7871-857-0.

No 462 **Lars Degerstedt:** Tabulation-based Logic Programming: A Multi-Level View of Query Answering, 1996, ISBN 91-7871-858-9.

No 475 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av hur ekonomiska styrsystem utformas och används efter företagsförvärv, 1997, ISBN 91-7871-914-3.

No 480 **Mikael Lindvall:** An Empirical Study of Requirements-Driven Impact Analysis in Object-Oriented Software Evolution, 1997, ISBN 91-7871-927-5.

No 485 **Göran Forslund:** Opinion-Based Systems: The Cooperative Perspective on Knowledge-Based Decision Support, 1997, ISBN 91-7871-938-0.

No 494 **Martin Sköld:** Active Database Management Systems for Monitoring and Control, 1997, ISBN 91-7219-002-7.

No 495 **Hans Olsén:** Automatic Verification of Petri Nets in a CLP framework, 1997, ISBN 91-7219-011-6.

No 498 **Thomas Drakengren:** Algorithms and Complexity for Temporal and Spatial Formalisms, 1997, ISBN 91-7219-019-1.

No 502 **Jakob Axelsson:** Analysis and Synthesis of Heterogeneous Real-Time Systems, 1997, ISBN 91-7219-035-3.

No 503 **Johan Ringström:** Compiler Generation for Data-Parallel Programming Languages from Two-Level Semantics Specifications, 1997, ISBN 91-7219-045-0.

No 512 **Anna Moberg:** Närhet och distans - Studier av kommunikationsmönster i satellitkontor och flexibla kontor, 1997, ISBN 91-7219-119-8.

No 520 **Mikael Ronström:** Design and Modelling of a Parallel Data Server for Telecom Applications, 1998, ISBN 91-7219-169-4.

No 522 **Niclas Ohlsson:** Towards Effective Fault Prevention - An Empirical Study in Software Engineering, 1998, ISBN 91-7219-176-7.

No 526 **Joachim Karlsson:** A Systematic Approach for Prioritizing Software Requirements, 1998, ISBN 91-7219-184-8.

No 530 **Henrik Nilsson:** Declarative Debugging for Lazy Functional Languages, 1998, ISBN 91-7219-197-x.

No 555 **Jonas Hallberg:** Timing Issues in High-Level Synthesis, 1998, ISBN 91-7219-369-7.

No 561 **Ling Lin:** Management of 1-D Sequence Data - From Discrete to Continuous, 1999, ISBN 91-7219-402-2.

No 563 **Eva L Ragnemalm:** Student Modelling based on Collaborative Dialogue with a Learning Companion, 1999, ISBN 91-7219-412-X.

No 567 **Jörgen Lindström:** Does Distance matter? On geographical dispersion in organisations, 1999, ISBN 91-7219-439-1.

No 582 **Vanja Josifovski:** Design, Implementation and Evaluation of a Distributed Mediator System for Data Integration, 1999, ISBN 91-7219-482-0.

No 589 **Rita Kovordányi:** Modeling and Simulating Inhibitory Mechanisms in Mental Image Reinterpretation - Towards Cooperative Human-Computer Creativity, 1999, ISBN 91-7219-506-1.

No 592 **Mikael Ericsson:** Supporting the Use of Design Knowledge - An Assessment of Commenting Agents, 1999, ISBN 91-7219-532-0.

No 593 **Lars Karlsson:** Actions, Interactions and Narratives, 1999, ISBN 91-7219-534-7.

No 594 **C. G. Mikael Johansson:** Social and Organizational Aspects of Requirements Engineering Methods - A practice-oriented approach, 1999, ISBN 91-7219-541-X.

No 595 **Jörgen Hansson:** Value-Driven Multi-Class Overload Management in Real-Time Database Systems, 1999, ISBN 91-7219-542-8.

No 596 **Niklas Hallberg:** Incorporating User Values in the Design of Information Systems and Services in the Public Sector: A Methods Approach, 1999, ISBN 91-7219-543-6.

No 597 **Vivian Vimarlund:** An Economic Perspective on the Analysis of Impacts of Information Technology: From Case Studies in Health-Care towards General Models and Theories, 1999, ISBN 91-7219-544-4.

No 598 **Johan Jenvald:** Methods and Tools in Computer-Supported Taskforce Training, 1999, ISBN 91-7219-547-9.

No 607 **Magnus Merkel:** Understanding and enhancing translation by parallel text processing, 1999, ISBN 91-7219-614-9.

No 611 **Silvia Coradeschi:** Anchoring symbols to sensory data, 1999, ISBN 91-7219-623-8.

No 613 **Man Lin:** Analysis and Synthesis of Reactive Systems: A Generic Layered Architecture Perspective, 1999, ISBN 91-7219-630-0.

No 618 **Jimmy Tjäder:** Systemimplementering i praktiken - En studie av logiker i fyra projekt, 1999, ISBN 91-7219-657-2.

No 627 **Vadim Engelson:** Tools for Design, Interactive Simulation, and Visualization of Object-Oriented Models in Scientific Computing, 2000, ISBN 91-7219-709-9.

No 637 **Esa Falkenroth:** Database Technology for Control and Simulation, 2000, ISBN 91-7219-766-8.

No 639 **Per-Arne Persson:** Bringing Power and Knowledge Together: Information Systems Design for Autonomy and Control in Command Work, 2000, ISBN 91-7219-796-X.

No 660 **Erik Larsson:** An Integrated System-Level Design for Testability Methodology, 2000, ISBN 91-7219-890-7.

No 688 **Marcus Bjäreland:** Model-based Execution Monitoring, 2001, ISBN 91-7373-016-5.

No 689 **Joakim Gustafsson:** Extending Temporal Action Logic, 2001, ISBN 91-7373-017-3.

No 720 **Carl-Johan Petri:** Organizational Information Provision - Managing Mandatory and Discretionary Use of Information Technology, 2001, ISBN-91-7373-126-9.

No 724 **Paul Scerri:** Designing Agents for Systems with Adjustable Autonomy, 2001, ISBN 91 7373 207 9.

No 725 **Tim Heyer:** Semantic Inspection of Software Artifacts: From Theory to Practice, 2001, ISBN 91 7373 208 7.

No 726 **Pär Carlshamre:** A Usability Perspective on Requirements Engineering - From Methodology to Product Development, 2001, ISBN 91 7373 212 5.

No 732 **Juha Takkinen:** From Information Management to Task Management in Electronic Mail, 2002, ISBN 91 7373 258 3.

No 745 **Johan Åberg:** Live Help Systems: An Approach to Intelligent Help for Web Information Systems, 2002, ISBN 91-7373-311-3.

No 746 **Rego Granlund:** Monitoring Distributed Teamwork Training, 2002, ISBN 91-7373-312-1.

No 757 **Henrik André-Jönsson:** Indexing Strategies for Time Series Data, 2002, ISBN 917373-346-6.

No 747 **Anneli Hagdahl:** Development of IT-supported Interorganisational Collaboration - A Case Study in the Swedish Public Sector, 2002, ISBN 91-7373-314-8.

No 749 **Sofie Pilemalm:** Information Technology for Non-Profit Organisations - Extended Participatory Design of an Information System for Trade Union Shop Stewards, 2002, ISBN 91-7373-318-0.

No 765 **Stefan Holmlid:** Adapting users: Towards a theory of use quality, 2002, ISBN 91-7373-397-0.

No 771 **Magnus Morin:** Multimedia Representations of Distributed Tactical Operations, 2002, ISBN 91-7373-421-7.

No 772 **Pawel Pietrzak:** A Type-Based Framework for Locating Errors in Constraint Logic Programs, 2002, ISBN 91-7373-422-5.

No 758 **Erik Berglund:** Library Communication Among Programmers Worldwide, 2002, ISBN 91-7373-349-0.

No 774 **Choong-ho Yi:** Modelling Object-Oriented Dynamic Systems Using a Logic-Based Framework, 2002, ISBN 91-7373-424-1.

No 779 **Mathias Broxvall:** A Study in the Computational Complexity of Temporal Reasoning, 2002, ISBN 91-7373-440-3.

No 793 **Asmus Pandikow:** A Generic Principle for Enabling Interoperability of Structured and Object-Oriented Analysis and Design Tools, 2002, ISBN 91-7373-479-9.

No 785 **Lars Hult:** Publika Informationstjänster. En studie av den Internetbaserade encyklopedins bruksegenskaper, 2003, ISBN 91-7373-461-6.

No 800 **Lars Taxén:** A Framework for the Coordination of Complex Systems´ Development, 2003, ISBN 91-7373-604-X

No 808 **Klas Gäre:** Tre perspektiv på förväntningar och förändringar i samband med införande av informationssystem, 2003, ISBN 91-7373-618-X.

No 821 **Mikael Kindborg:** Concurrent Comics - programming of social agents by children, 2003, ISBN 91-7373-651-1.

No 823 **Christina Ölvingson:** On Development of Information Systems with GIS Functionality in Public Health Informatics: A Requirements Engineering Approach, 2003, ISBN 91-7373-656-2.

No 828 **Tobias Ritzau:** Memory Efficient Hard Real-Time Garbage Collection, 2003, ISBN 91-7373-666-X.

No 833 **Paul Pop:** Analysis and Synthesis of Communication-Intensive Heterogeneous Real-Time Systems, 2003, ISBN 91-7373-683-X.

No 852 **Johan Moe:** Observing the Dynamic Behaviour of Large Distributed Systems to Improve Development and Testing – An Empirical Study in Software Engineering, 2003, ISBN 91-7373-779-8.

No 867 **Erik Herzog:** An Approach to Systems Engineering Tool Data Representation and Exchange, 2004, ISBN 91-7373-929-4.

No 872 **Aseel Berglund:** Augmenting the Remote Control: Studies in Complex Information Navigation for Digital TV, 2004, ISBN 91-7373-940-5.

No 869 **Jo Skåmedal:** Telecommuting's Implications on Travel and Travel Patterns, 2004, ISBN 91-7373-935-9.

No 870 **Linda Askenäs:** The Roles of IT - Studies of Organising when Implementing and Using Enterprise Systems, 2004, ISBN 91-7373-936-7.

No 874 **Annika Flycht-Eriksson:** Design and Use of Ontologies in Information-Providing Dialogue Systems, 2004, ISBN 91-7373-947-2.

No 873 **Peter Bunus:** Debugging Techniques for Equation-Based Languages, 2004, ISBN 91-7373-941-3.

No 876 **Jonas Mellin:** Resource-Predictable and Efficient Monitoring of Events, 2004, ISBN 91-7373-956-1.

No 883 **Magnus Bång:** Computing at the Speed of Paper: Ubiquitous Computing Environments for Healthcare Professionals, 2004, ISBN 91-7373-971-5

No 882 **Robert Eklund:** Disfluency in Swedish human-human and human-machine travel booking dialogues, 2004, ISBN 91-7373-966-9.

No 887 **Anders Lindström:** English and other Foreign Linguistic Elements in Spoken Swedish. Studies of Productive Processes and their Modelling using Finite-State Tools, 2004, ISBN 91-7373-981-2.

No 889 **Zhiping Wang:** Capacity-Constrained Production-inventory systems - Modelling and Analysis in both a traditional and an e-business context, 2004, ISBN 91-85295-08-6.

No 893 **Pernilla Qvarfordt:** Eyes on Multimodal Interaction, 2004, ISBN 91-85295-30-2.

No 910 **Magnus Kald:** In the Borderland between Strategy and Management Control - Theoretical Framework and Empirical Evidence, 2004, ISBN 91-85295-82-5.

No 918 **Jonas Lundberg:** Shaping Electronic News: Genre Perspectives on Interaction Design, 2004, ISBN 91-85297-14-3.

No 900 **Mattias Arvola:** Shades of use: The dynamics of interaction design for sociable use, 2004, ISBN 91-85295-42-6.

No 920 **Luis Alejandro Cortés:** Verification and Scheduling Techniques for Real-Time Embedded Systems, 2004, ISBN 91-85297-21-6.

No 929 **Diana Szentivanyi:** Performance Studies of Fault-Tolerant Middleware, 2005, ISBN 91-85297-58-5.

No 933 **Mikael Cäker:** Management Accounting as Constructing and Opposing Customer Focus: Three Case Studies on Management Accounting and Customer Relations, 2005, ISBN 91-85297-64-X.

No 937 **Jonas Kvarnström:** TALplanner and Other Extensions to Temporal Action Logic, 2005, ISBN 91-85297-75-5.

No 938 **Bourhane Kadmiry:** Fuzzy Gain-Scheduled Visual Servoing for Unmanned Helicopter, 2005, ISBN 91-85297-76-3.

No 945 **Gert Jervan:** Hybrid Built-In Self-Test and Test Generation Techniques for Digital Systems, 2005, ISBN: 91-85297-97-6.

No 946 **Anders Arpteg:** Intelligent Semi-Structured Information Extraction, 2005, ISBN 91-85297-98-4.

No 947 **Ola Angelsmark:** Constructing Algorithms for Constraint Satisfaction and Related Problems - Methods and Applications, 2005, ISBN 91-85297-99-2.

No 963 **Calin Curescu:** Utility-based Optimisation of Resource Allocation for Wireless Networks, 2005, ISBN 91-85457-07-8.

No 972 **Björn Johansson:** Joint Control in Dynamic Situations, 2005, ISBN 91-85457-31-0.

No 974 **Dan Lawesson:** An Approach to Diagnosability Analysis for Interacting Finite State Systems, 2005, ISBN 91-85457-39-6.

No 979 **Claudiu Duma:** Security and Trust Mechanisms for Groups in Distributed Services, 2005, ISBN 91-85457-54-X.

No 983 **Sorin Manolache:** Analysis and Optimisation of Real-Time Systems with Stochastic Behaviour, 2005, ISBN 91-85457-60-4.

No 986 **Yuxiao Zhao:** Standards-Based Application Integration for Business-to-Business Communications, 2005, ISBN 91-85457-66-3.

No 1004 **Patrik Haslum:** Admissible Heuristics for Automated Planning, 2006, ISBN 91-85497-28-2.

No 1005 **Aleksandra Tešanovic:** Developing Reusable and Reconfigurable Real-Time Software using Aspects and Components, 2006, ISBN 91-85497-29-0.

No 1008 **David Dinka:** Role, Identity and Work: Extending the design and development agenda, 2006, ISBN 91-85497-42-8.

No 1009 **Iakov Nakhimovski:** Contributions to the Modeling and Simulation of Mechanical Systems with Detailed Contact Analysis, 2006, ISBN 91-85497-43-X.

No 1013 **Wilhelm Dahllöf:** Exact Algorithms for Exact Satisfiability Problems, 2006, ISBN 91-85523-97-6.

No 1016 **Levon Saldamli:** PDEModelica - A High-Level Language for Modeling with Partial Differential Equations, 2006, ISBN 91-85523-84-4.

No 1017 **Daniel Karlsson:** Verification of Component-based Embedded System Designs, 2006, ISBN 91-85523-79-8

No 1018 **Ioan Chisalita:** Communication and Networking Techniques for Traffic Safety Systems, 2006, ISBN 91-85523-77-1.

No 1019 **Tarja Susi:** The Puzzle of Social Activity - The Significance of Tools in Cognition and Cooperation, 2006, ISBN 91-85523-71-2.

No 1021 **Andrzej Bednarski:** Integrated Optimal Code Generation for Digital Signal Processors, 2006, ISBN 91-85523-69-0.

No 1022 **Peter Aronsson:** Automatic Parallelization of Equation-Based Simulation Programs, 2006, ISBN 91-85523-68-2.

No 1030 **Robert Nilsson:** A Mutation-based Framework for Automated Testing of Timeliness, 2006, ISBN 91-85523-35-6.

No 1034 **Jon Edvardsson:** Techniques for Automatic Generation of Tests from Programs and Specifications, 2006, ISBN 91-85523-31-3.

No 1035 **Vaida Jakoniene:** Integration of Biological Data, 2006, ISBN 91-85523-28-3.

No 1045 **Genevieve Gorrell:** Generalized Hebbian Algorithms for Dimensionality Reduction in Natural Language Processing, 2006, ISBN 91-85643-88-2.

No 1051 **Yu-Hsing Huang:** Having a New Pair of Glasses - Applying Systemic Accident Models on Road Safety, 2006, ISBN 91-85643-64-5.

No 1054 **Åsa Hedenskog:** Perceive those things which cannot be seen - A Cognitive Systems Engineering perspective on requirements management, 2006, ISBN 91-85643-57-2.

No 1061 **Cécile Åberg:** An Evaluation Platform for Semantic Web Technology, 2007, ISBN 91-85643-31-9.

No 1073 **Mats Grindal:** Handling Combinatorial Explosion in Software Testing, 2007, ISBN 978-91-85715-74-9.

No 1075 **Almut Herzog:** Usable Security Policies for Runtime Environments, 2007, ISBN 978-91-85715-65-7.

No 1079 **Magnus Wahlström:** Algorithms, measures, and upper bounds for Satisfiability and related problems, 2007, ISBN 978-91-85715-55-8.

No 1083 **Jesper Andersson:** Dynamic Software Architectures, 2007, ISBN 978-91-85715-46-6.

No 1086 **Ulf Johansson:** Obtaining Accurate and Comprehensible Data Mining Models - An Evolutionary Approach, 2007, ISBN 978-91-85715-34-3.

No 1089 **Traian Pop:** Analysis and Optimisation of Distributed Embedded Systems with Heterogeneous Scheduling Policies, 2007, ISBN 978-91-85715-27-5.

No 1091 **Gustav Nordh:** Complexity Dichotomies for CSP-related Problems, 2007, ISBN 978-91-85715-20-6.

No 1106 **Per Ola Kristensson:** Discrete and Continuous Shape Writing for Text Entry and Control, 2007, ISBN 978-91-85831-77-7.

No 1110 **He Tan:** Aligning Biomedical Ontologies, 2007, ISBN 978-91-85831-56-2.

No 1112 **Jessica Lindblom:** Minding the body - Interacting socially through embodied action, 2007, ISBN 978-91-85831-48-7.

No 1113 **Pontus Wärnestål:** Dialogue Behavior Management in Conversational Recommender Systems, 2007, ISBN 978-91-85831-47-0.

No 1120 **Thomas Gustafsson:** Management of Real-Time Data Consistency and Transient Overloads in Embedded Systems, 2007, ISBN 978-91-85831-33-3.

No 1127 **Alexandru Andrei:** Energy Efficient and Predictable Design of Real-time Embedded Systems, 2007, ISBN 978-91-85831-06-7.

No 1139 **Per Wikberg:** Eliciting Knowledge from Experts in Modeling of Complex Systems: Managing Variation and Interactions, 2007, ISBN 978-91-85895-66-3.

No 1143 **Mehdi Amirijoo:** QoS Control of Real-Time Data Services under Uncertain Workload, 2007, ISBN 978-91-85895-49-6.

No 1150 **Sanny Syberfeldt:** Optimistic Replication with Forward Conflict Resolution in Distributed Real-Time Databases, 2007, ISBN 978-91-85895-27-4.

No 1155 **Beatrice Alenljung:** Envisioning a Future Decision Support System for Requirements Engineering - A Holistic and Human-centred Perspective, 2008, ISBN 978-91-85895-11-3.

No 1156 **Artur Wilk:** Types for XML with Application to Xcerpt, 2008, ISBN 978-91-85895-08-3.

No 1183 **Adrian Pop:** Integrated Model-Driven Development Environments for Equation-Based Object-Oriented Languages, 2008, ISBN 978-91-7393-895-2.

No 1185 **Jörgen Skågeby:** Gifting Technologies - Ethnographic Studies of End-users and Social Media Sharing, 2008, ISBN 978-91-7393-892-1.

No 1187 **Imad-Eldin Ali Abugessaisa:** Analytical tools and information-sharing methods supporting road safety organizations, 2008, ISBN 978-91-7393-887-7.

No 1204 **H. Joe Steinhauer:** A Representation Scheme for Description and Reconstruction of Object Configurations Based on Qualitative Relations, 2008, ISBN 978-91-7393-823-5.

No 1222 **Anders Larsson:** Test Optimization for Core-based System-on-Chip, 2008, ISBN 978-91-7393-768-9.

No 1238 **Andreas Borg:** Processes and Models for Capacity Requirements in Telecommunication Systems, 2009, ISBN 978-91-7393-700-9.

No 1240 **Fredrik Heintz:** DyKnow: A Stream-Based Knowledge Processing Middleware Framework, 2009, ISBN 978-91-7393-696-5.

No 1241 **Birgitta Lindström:** Testability of Dynamic Real-Time Systems, 2009, ISBN 978-91-7393-695-8.

No 1244 **Eva Blomqvist:** Semi-automatic Ontology Construction based on Patterns, 2009, ISBN 978-91-7393-683-5.

No 1249 **Rogier Woltjer:** Functional Modeling of Constraint Management in Aviation Safety and Command and Control, 2009, ISBN 978-91-7393-659-0.

No 1260 **Gianpaolo Conte:** Vision-Based Localization and Guidance for Unmanned Aerial Vehicles, 2009, ISBN 978-91-7393-603-3.

No 1262 **AnnMarie Ericsson:** Enabling Tool Support for Formal Analysis of ECA Rules, 2009, ISBN 978-91-7393-598-2.

No 1266 **Jiri Trnka:** Exploring Tactical Command and Control: A Role-Playing Simulation Approach, 2009, ISBN 978-91-7393-571-5.

No 1268 **Bahlol Rahimi:** Supporting Collaborative Work through ICT - How End-users Think of and Adopt Integrated Health Information Systems, 2009, ISBN 978-91-7393-550-0.

No 1274 **Fredrik Kuivinen:** Algorithms and Hardness Results for Some Valued CSPs, 2009, ISBN 978-91-7393-525-8.

Linköping Studies in Statistics

No 9 **Davood Shahsavani:** Computer Experiments Designed to Explore and Approximate Complex Deterministic Models, 2008, ISBN 978-91-7393-976-8.

No 10 **Karl Wahlin:** Roadmap for Trend Detection and Assessment of Data Quality, 2008, ISBN 978-91-7393-792-4

Linköping Studies in Information Science

No 1 **Karin Axelsson:** Metodisk systemstrukturering- att skapa samstämmighet mellan informationssystemarkitektur och verksamhet, 1998. ISBN-9172-19-296-8.

No 2 **Stefan Cronholm:** Metodverktyg och användbarhet - en studie av datorstödd metodbaserad systemutveckling, 1998, ISBN-9172-19-299-2.

No 3 **Anders Avdic:** Användare och utvecklare - om anveckling med kalkylprogram, 1999. ISBN-91-7219-606-8.

No 4 **Owen Eriksson:** Kommunikationskvalitet hos informationssystem och affärsprocesser, 2000, ISBN 91-7219-811-7.

No 5 **Mikael Lind:** Från system till process - kriterier för processbestämning vid verksamhetsanalys, 2001, ISBN 91-7373-067-X.

No 6 **Ulf Melin:** Koordination och informationssystem i företag och nätverk, 2002, ISBN 91-7373-278-8.

No 7 **Pär J. Ågerfalk:** Information Systems Actability - Understanding Information Technology as a Tool for Business Action and Communication, 2003, ISBN 91-7373-628-7.

No 8 **Ulf Seigerroth:** Att förstå och förändra systemutvecklingsverksamheter - en taxonomi för metautveckling, 2003, ISBN91-7373-736-4.

No 9 **Karin Hedström:** Spår av datoriseringens värden – Effekter av IT i äldreomsorg, 2004, ISBN 91-7373-963-4.

No 10 **Ewa Braf:** Knowledge Demanded for Action - Studies on Knowledge Mediation in Organisations, 2004, ISBN 91-85295-47-7.

No 11 **Fredrik Karlsson:** Method Configuration method and computerized tool support, 2005, ISBN 91-85297-48-8.

No 12 **Malin Nordström:** Styrbar systemförvaltning - Att organisera systemförvaltningsverksamhet med hjälp av effektiva förvaltningsobjekt, 2005, ISBN 91-85297-60-7.

No 13 **Stefan Holgersson:** Yrke: POLIS - Yrkeskunskap, motivation, IT-system och andra förutsättningar för polisarbete, 2005, ISBN 91-85299-43-X.

No 14 **Benneth Christiansson, Marie-Therese Christiansson:** Mötet mellan process och komponent - mot ett ramverk för en verksamhetsnära kravspecifikation vid anskaffning av komponentbaserade informationssystem, 2006, ISBN 91-85643-22-X.